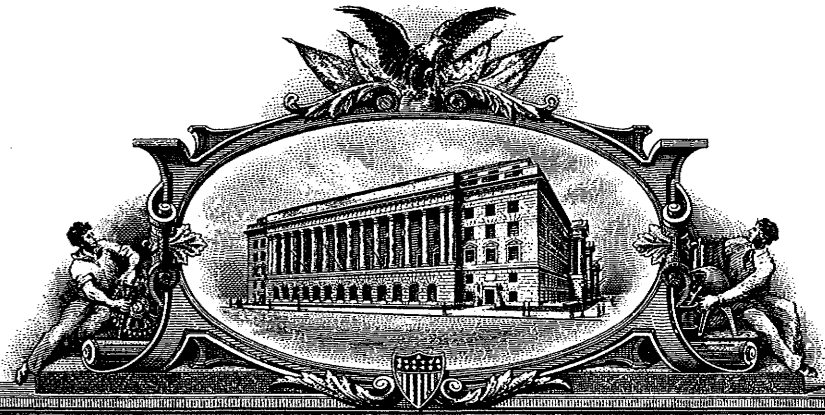


IW 7535866



THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 08, 2015

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE
RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS
OF:

APPLICATION NUMBER: 09/629,576

FILING DATE: July 31, 2000

PATENT NUMBER: 6,829,634

ISSUE DATE: December 07, 2004

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. LAWRENCE
Certifying Officer



1c869 U.S. PTO
09/629576
07/31/00

709	204
Class	Subclass
ISSUE CLASSIFICATION	

PATENT NUMBER
6829634
6829634

U.S. UTILITY Patent Application

U.P.F. SCANNED	U.I.P.E. Q.A.	PATENT DATE DEC 0 2004
-------------------	------------------	---------------------------

APPLICATION NO. 09/629576	CONT/PRIOR	CLASS 370 709	SUBCLASS 204	ART UNIT 2664 2155	EXAMINER Wen Young
------------------------------	------------	--------------------------------	-----------------	-------------------------------------	-----------------------

APPLICANTS
Fred Holt
Virgil Bourassa

Broadcasting network

TITLE

PTO-2040
12/99

ISSUING CLASSIFICATION									
ORIGINAL			CROSS REFERENCE(S)						
CLASS	SUBCLASS		CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
709	204		709	205	203	243	201	228	319
INTERNATIONAL CLASSIFICATION				225					
G06F	15/16		370	236					

Continued on Issue Slip Inside File Jacket

<input type="checkbox"/> TERMINAL DISCLAIMER	DRAWINGS			CLAIMS ALLOWED	
	Sheets Drwg. 39	Figs. Drwg. 4/44	Print Fig. 1	Total Claims 24	Print Claim for O.G. 1
<input type="checkbox"/> The term of this patent subsequent to _____ (date) has been disclaimed.	Duke Sh (Assistant Examiner)		7/7/04 (Date)	NOTICE OF ALLOWANCE MAILED 7/13/04	
<input type="checkbox"/> The term of this patent shall not extend beyond the expiration date of U.S. Patent. No. _____	HOSAIN ALAM SUPERVISORY PATENT EXAMINER Molen (Primary Examiner)			7/15/04 (Date)	
<input type="checkbox"/> The terminal _____ months of this patent have been disclaimed.	L. Johnson (Legal Instruments Examiner)		7-15-04 (Date)	ISSUE FEE Amount Due \$1330 ⁰⁰ Date Paid 7/15/04 JPM	
WARNING: The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.			ISSUE BATCH NUMBER		

Form PTO-435A
(Rev. 6/99)

FILED WITH: DISK (CRF) FICHE CD-ROM
(Attached in pocket on right inside flap)

ISSUE FEE IN FILE

(FACE)

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

PTO-1556
(5/87)

*U.S. GPO: 1969-458-082/10144

Please type a plus sign (+) inside this box

+

PTO/SB/05 (4/98)

UTILITY

PATENT APPLICATION TRANSMITTAL

Attorney Docket No. 030048001

First Inventor or Application Identifier Virgil E. Bourassa

Title BROADCASTING NETWORK

Express Mail Label No. EL404935384US

(Only for nonprovisional applications under 37 CFR § 1.53(b))

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO:

Box Patent Application
Assistant Commissioner for Patents
Washington, D.C. 20231

- 1. Authorization for Extensions & Fee Transmittal (Submit an original and a duplicate for fee processing)
- 2. Specification [Total Pages] 46
(preferred arrangement set forth below)
 - Descriptive Title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure

- 5. Microfiche Computer Program (Appendix)
- 6. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
 - a. Computer-Readable Copy
 - b. Paper Copy (identical to computer copy)
 - c. Statement verifying identity of above copies

- 3. Drawing(s) (35 USC 113) [Total Sheets] 39
- 4. Oath or Declaration [Total Pages]
 - a. Newly executed (original or copy)
 - b. Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with Box 16 completed)
 - i. DELETION OF INVENTOR(S)
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b)

ACCOMPANYING APPLICATION PARTS

- 7. Assignment Papers (cover sheet & document(s))
- 8. 37 CFR 3.73(b) Statement Power of Attorney (when there is an assignee)
- 9. English Translation Document (if applicable)
- 10. Information Disclosure Statement (IDS)/PTO-1449 Copies of IDS Citations
- 11. Preliminary Amendment
- 12. Return Receipt Postcard
- 12. Small Entity Statement filed in prior application, Status still proper and desired
- 14. Certified Copy of Priority Document(s) (if foreign priority is claimed)
- 15. Other: _____

16. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information below and in a preliminary amendment

Continuation Divisional Continuation-In-Part (CIP) of prior Application No.: _____

Prior application information: Examiner _____ Group / Art Unit _____

For CONTINUATION or DIVISIONAL apps only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

Claims the benefit of Provisional Application No. _____

17. CORRESPONDENCE ADDRESS

Customer Number 25096 / Barcode

Respectfully submitted,

TYPED or PRINTED NAME Maurice J. Pirio

REGISTRATION NO. 33,273

SIGNATURE Maurice J. Pirio

Date July 31, 2000

BROADCASTING NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to U.S. Patent Application No. _____,
 entitled "BROADCASTING NETWORK," filed on July 31, 2000 (Attorney Docket
 No. 030048001 US); U.S. Patent Application No. _____, entitled "JOINING A
 BROADCAST CHANNEL," filed on July 31, 2000 (Attorney Docket No. 030048002 US);
 U.S. Patent Application No. _____, "LEAVING A BROADCAST CHANNEL,"
 filed on July 31, 2000 (Attorney Docket No. 030048003 US); U.S. Patent Application
 No. _____, entitled "BROADCASTING ON A BROADCAST CHANNEL," filed
 on July 31, 2000 (Attorney Docket No. 030048004 US); U.S. Patent Application
 No. _____, entitled "CONTACTING A BROADCAST CHANNEL," filed on
 July 31, 2000 (Attorney Docket No. 030048005 US); U.S. Patent Application
 No. _____, entitled "DISTRIBUTED AUCTION SYSTEM," filed on
 July 31, 2000 (Attorney Docket No. 030048006 US); U.S. Patent Application
 No. _____, entitled "AN INFORMATION DELIVERY SERVICE," filed on
 July 31, 2000 (Attorney Docket No. 030048007 US); U.S. Patent Application
 No. _____, entitled "DISTRIBUTED CONFERENCING SYSTEM," filed on
 July 31, 2000 (Attorney Docket No. 030048008 US); and U.S. Patent Application
 No. _____, entitled "DISTRIBUTED GAME ENVIRONMENT," filed on
 July 31, 2000 (Attorney Docket No. 030048009 US), the disclosures of which are
 incorporated herein by reference.

TECHNICAL FIELD

The described technology relates generally to a computer network and more particularly, to a broadcast channel for a subset of a computers of an underlying network.

BACKGROUND

There are a wide variety of computer network communications techniques such as point-to-point network protocols, client/server middleware, multicasting network

protocols, and peer-to-peer middleware. Each of these communications techniques have their advantages and disadvantages, but none is particularly well suited to the simultaneous sharing of information among computers that are widely distributed. For example, collaborative processing applications, such as a network meeting programs, have a need to
5 distribute information in a timely manner to all participants who may be geographically distributed.

The point-to-point network protocols, such as UNIX pipes, TCP/IP, and UDP, allow processes on different computers to communicate via point-to-point connections. The interconnection of all participants using point-to-point connections, while theoretically
10 possible, does not scale well as a number of participants grows. For example, each participating process would need to manage its direct connections to all other participating processes. Programmers, however, find it very difficult to manage single connections, and management of multiple connections is much more complex. In addition, participating processes may be limited to the number of direct connections that they can support. This
15 limits the number of possible participants in the sharing of information.

The client/server middleware systems provide a server that coordinates the communications between the various clients who are sharing the information. The server functions as a central authority for controlling access to shared resources. Examples of client/server middleware systems include remote procedure calls ("RPC"), database servers, and the common object request broker architecture ("CORBA"). Client/server middleware systems are not particularly well suited to sharing of information among many participants. In particular, when a client stores information to be shared at the server, each other client
20 would need to poll the server to determine that new information is being shared. Such polling places a very high overhead on the communications network. Alternatively, each client may register a callback with the server, which the server then invokes when new
25 information is available to be shared. Such a callback technique presents a performance bottleneck because a single server needs to call back to each client whenever new information is to be shared. In addition, the reliability of the entire sharing of information depends upon the reliability of the single server. Thus, a failure at a single computer (*i.e.*,
30 the server) would prevent communications between any of the clients.

The multicasting network protocols allow the sending of broadcast messages to multiple recipients of a network. The current implementations of such multicasting network

protocols tend to place an unacceptable overhead on the underlying network. For example, UDP multicasting would swamp the Internet when trying to locate all possible participants. IP multicasting has other problems that include needing special-purpose infrastructure (e.g., routers) to support the sharing of information efficiently.

5 The peer-to-peer middleware communications systems rely on a multicasting network protocol or a graph of point-to-point network protocols. Such peer-to-peer middleware is provided by the T.120 Internet standard, which is used in such products as Data Connection's D.C.-share and Microsoft's NetMeeting. These peer-to-peer middleware systems rely upon a user to assemble a point-to-point graph of the connections used for
10 sharing the information. Thus, it is neither suitable nor desirable to use peer-to-peer middleware systems when more than a small number of participants is desired. In addition, the underlying architecture of the T.120 Internet standard is a tree structure, which relies on the root node of the tree for reliability of the entire network. That is, each message must pass through the root node in order to be received by all participants.

15 It would be desirable to have a reliable communications network that is suitable for the simultaneous sharing of information among a large number of the processes that are widely distributed.

20
25
30
BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a graph that is 4-regular and 4-connected which represents a broadcast channel.

Figure 2 illustrates a graph representing 20 computers connected to a broadcast channel.

Figures 3A and 3B illustrate the process of connecting a new computer Z to the broadcast channel.

25 Figure 4A illustrates the broadcast channel of Figure 1 with an added computer.

Figure 4B illustrates the broadcast channel of Figure 4A with an added computer.

30 Figure 4C also illustrates the broadcast channel of Figure 4A with an added computer.

Figure 5A illustrates the disconnecting of a computer from the broadcast channel in a planned manner.

Figure 5B illustrates the disconnecting of a computer from the broadcast channel in an unplanned manner.

5 Figure 5C illustrates the neighbors with empty ports condition.

Figure 5D illustrates two computers that are not neighbors who now have empty ports.

Figure 5E illustrates the neighbors with empty ports condition in the small regime.

10 Figure 5F illustrates the situation of Figure 5E when in the large regime.

Figure 6 is a block diagram illustrating components of a computer that is connected to a broadcast channel.

Figure 7 is a block diagram illustrating the sub-components of the broadcaster component in one embodiment.

15 Figure 8 is a flow diagram illustrating the processing of the connect routine in one embodiment.

Figure 9 is a flow diagram illustrating the processing of the seek portal computer routine in one embodiment.

20 Figure 10 is a flow diagram illustrating the processing of the contact process routine in one embodiment.

Figure 11 is a flow diagram illustrating the processing of the connect request routine in one embodiment.

Figure 12 is a flow diagram of the processing of the check for external call routine in one embodiment.

25 Figure 13 is a flow diagram of the processing of the achieve connection routine in one embodiment.

Figure 14 is a flow diagram illustrating the processing of the external dispatcher routine in one embodiment.

30 Figure 15 is a flow diagram illustrating the processing of the handle seeking connection call routine in one embodiment.

Figure 16 is a flow diagram illustrating processing of the handle connection request call routine in one embodiment.

Figure 17 is a flow diagram illustrating the processing of the add neighbor routine in one embodiment.

Figure 18 is a flow diagram illustrating the processing of the forward connection edge search routine in one embodiment.

5 Figure 19 is a flow diagram illustrating the processing of the handle edge proposal call routine.

Figure 20 is a flow diagram illustrating the processing of the handle port connection call routine in one embodiment.

10 Figure 21 is a flow diagram illustrating the processing of the fill hole routine in one embodiment.

Figure 22 is a flow diagram illustrating the processing of the internal dispatcher routine in one embodiment.

Figure 23 is a flow diagram illustrating the processing of the handle broadcast message routine in one embodiment.

15 Figure 24 is a flow diagram illustrating the processing of the distribute broadcast message routine in one embodiment.

Figure 26 is a flow diagram illustrating the processing of the handle connection port search statement routine in one embodiment.

20 Figure 27 is a flow diagram illustrating the processing of the court neighbor routine in one embodiment.

Figure 28 is a flow diagram illustrating the processing of the handle connection edge search call routine in one embodiment.

Figure 29 is a flow diagram illustrating the processing of the handle connection edge search response routine in one embodiment.

25 Figure 30 is a flow diagram illustrating the processing of the broadcast routine in one embodiment.

Figure 31 is a flow diagram illustrating the processing of the acquire message routine in one embodiment.

30 Figure 32 is a flow diagram illustrating processing of the handle condition check message in one embodiment.

Figure 33 is a flow diagram illustrating processing of the handle condition repair statement routine in one embodiment.

Figure 34 is a flow diagram illustrating the processing of the handle condition double check routine.

DETAILED DESCRIPTION

A broadcast technique in which a broadcast channel overlays a point-to-point communications network is provided. The broadcasting of a message over the broadcast channel is effectively a multicast to those computers of the network that are currently connected to the broadcast channel. In one embodiment, the broadcast technique provides a logical broadcast channel to which host computers through their executing processes can be connected. Each computer that is connected to the broadcast channel can broadcast messages onto and receive messages off of the broadcast channel. Each computer that is connected to the broadcast channel receives all messages that are broadcast while it is connected. The logical broadcast channel is implemented using an underlying network system (e.g., the Internet) that allows each computer connected to the underlying network system to send messages to each other connected computer using each computer's address. Thus, the broadcast technique effectively provides a broadcast channel using an underlying network system that sends messages on a point-to-point basis.

The broadcast technique overlays the underlying network system with a graph of point-to-point connections (i.e., edges) between host computers (i.e., nodes) through which the broadcast channel is implemented. In one embodiment, each computer is connected to four other computers, referred to as neighbors. (Actually, a process executing on a computer is connected to four other processes executing on this or four other computers.) To broadcast a message, the originating computer sends the message to each of its neighbors using its point-to-point connections. Each computer that receives the message then sends the message to its three other neighbors using the point-to-point connections. In this way, the message is propagated to each computer using the underlying network to effect the broadcasting of the message to each computer over a logical broadcast channel. A graph in which each node is connected to four other nodes is referred to as a 4-regular graph. The use of a 4-regular graph means that a computer would become disconnected from the broadcast channel only if all four of the connections to its neighbors fail. The graph used by the broadcast technique also has the property that it would take a failure of four computers to

divide the graph into disjoint sub-graphs, that is two separate broadcast channels. This property is referred to as being 4-connected. Thus, the graph is both 4-regular and 4-connected.

Figure 1 illustrates a graph that is 4-regular and 4-connected which represents the broadcast channel. Each of the nine nodes A-I represents a computer that is connected to the broadcast channel, and each of the edges represents an "edge" connection between two computers of the broadcast channel. The time it takes to broadcast a message to each computer on the broadcast channel depends on the speed of the connections between the computers and the number of connections between the originating computer and each other computer on the broadcast channel. The minimum number of connections that a message would need to traverse between each pair of computers is the "distance" between the computers (*i.e.*, the shortest path between the two nodes of the graph). For example, the distance between computers A and F is one because computer A is directly connected to computer F. The distance between computers A and B is two because there is no direct connection between computers A and B, but computer F is directly connected to computer B. Thus, a message originating at computer A would be sent directly to computer F, and then sent from computer F to computer B. The maximum of the distances between the computers is the "diameter" of broadcast channel. The diameter of the broadcast channel represented by Figure 1 is two. That is, a message sent by any computer would traverse no more than two connections to reach every other computer. Figure 2 illustrates a graph representing 20 computers connected to a broadcast channel. The diameter of this broadcast channel is 4. In particular, the shortest path between computers 1 and 3 contains four connections (1-12, 12-15, 15-18, and 18-3).

The broadcast technique includes (1) the connecting of computers to the broadcast channel (*i.e.*, composing the graph), (2) the broadcasting of messages over the broadcast channel (*i.e.*, broadcasting through the graph), and (3) the disconnecting of computers from the broadcast channel (*i.e.*, decomposing the graph) composing the graph.

Composing the Graph

To connect to the broadcast channel, the computer seeking the connection first locates a computer that is currently fully connected to the broadcast channel and then

establishes a connection with four of the computers that are already connected to the broadcast channel. (This assumes that there are at least four computers already connected to the broadcast channel. When there are fewer than five computers connected, the broadcast channel cannot be a 4-regular graph. In such a case, the broadcast channel is considered to be in a "small regime." The broadcast technique for the small regime is described below in detail. When five or more computers are connected, the broadcast channel is considered to be in the "large regime." This description assumes that the broadcast channel is in the large regime, unless specified otherwise.) Thus, the process of connecting to the broadcast channel includes locating the broadcast channel, identifying the neighbors for the connecting computer, and then connecting to each identified neighbor. Each computer is aware of one or more "portal computers" through which that computer may locate the broadcast channel. A seeking computer locates the broadcast channel by contacting the portal computers until it finds one that is currently fully connected to the broadcast channel. The found portal computer then directs the identifying of four computers (*i.e.*, to be the seeking computer's neighbors) to which the seeking computer is to connect. Each of these four computers then cooperates with the seeking computer to effect the connecting of the seeking computer to the broadcast channel. A computer that has started the process of locating a portal computer, but does not yet have a neighbor, is in the "seeking connection state." A computer that is connected to at least one neighbor, but not yet four neighbors, is in the "partially connected state." A computer that is currently, or has been, previously connected to four neighbors is in the "fully connected state."

Since the broadcast channel is a 4-regular graph, each of the identified computers is already connected to four computers. Thus, some connections between computers need to be broken so that the seeking computer can connect to four computers. In one embodiment, the broadcast technique identifies two pairs of computers that are currently connected to each other. Each of these pairs of computers breaks the connection between them, and then each of the four computers (two from each pair) connects to the seeking computer. Figures 3A and 3B illustrate the process of a new computer Z connecting to the broadcast channel. Figure 3A illustrates the broadcast channel before computer Z is connected. The pairs of computers B and E and computers C and D are the two pairs that are identified as the neighbors for the new computer Z. The connections between each of these pairs is broken, and a connection between computer Z and each of computers B, C, D, and E

is established as indicated by Figure 3B. The process of breaking the connection between two neighbors and reconnecting each of the former neighbors to another computer is referred to as "edge pinning" as the edge between two nodes may be considered to be stretched and pinned to a new node.

5 Each computer connected to the broadcast channel allocates five communications ports for communicating with other computers. Four of the ports are referred to as "internal" ports because they are the ports through which the messages of the broadcast channels are sent. The connections between internal ports of neighbors are referred to as "internal" connections. Thus, the internal connections of the broadcast channel
10 form the 4-regular and 4-connected graph. The fifth port is referred to as an "external" port because it is used for sending non-broadcast messages between two computers. Neighbors can send non-broadcast messages either through their internal ports of their connection or through their external ports. A seeking computer uses external ports when locating a portal computer.

15 In one embodiment, the broadcast technique establishes the computer connections using the TCP/IP communications protocol, which is a point-to-point protocol, as the underlying network. The TCP/IP protocol provides for reliable and ordered delivery of messages between computers. The TCP/IP protocol provides each computer with a "port space" that is shared among all the processes that may execute on that computer. The ports are identified by numbers from 0 to 65,535. The first 2056 ports are reserved for specific applications (e.g., port 80 for HTTP messages). The remainder of the ports are user ports that are available to any process. In one embodiment, a set of port numbers can be reserved for use by the computer connected to the broadcast channel. In an alternative embodiment,
25 the port numbers used are dynamically identified by each computer. Each computer dynamically identifies an available port to be used as its call-in port. This call-in port is used to establish connections with the external port and the internal ports. Each computer that is connected to the broadcast channel can receive non-broadcast messages through its external port. A seeking computer tries "dialing" the port numbers of the portal computers until a portal computer "answers," a call on its call-in port. A portal computer answers when it is
30 connected to or attempting to connect to the broadcast channel and its call-in port is dialed. (In this description, a telephone metaphor is used to describe the connections.) When a computer receives a call on its call-in port, it transfers the call to another port. Thus, the

seeking computer actually communicates through that transfer-to port, which is the external port. The call is transferred so that other computers can place calls to that computer via the call-in port. The seeking computer then communicates via that external port to request the portal computer to assist in connecting the seeking computer to the broadcast channel. The seeking computer could identify the call-in port number of a portal computer by successively dialing each port in port number order. As discussed below in detail, the broadcast technique uses a hashing algorithm to select the port number order, which may result in improved performance.

A seeking computer could connect to the broadcast channel by connecting to computers either directly connected to the found portal computer or directly connected to one of its neighbors. A possible problem with such a scheme for identifying the neighbors for the seeking computer is that the diameter of the broadcast channel may increase when each seeking computer uses the same found portal computer and establishes a connection to the broadcast channel directly through that found portal computer. Conceptually, the graph becomes elongated in the direction of where the new nodes are added. Figures 4A-4C illustrate that possible problem. Figure 4A illustrates the broadcast channel of Figure 1 with an added computer. Computer J was connected to the broadcast channel by edge pinning edges C-D and E-H to computer J. The diameter of this broadcast channel is still two. Figure 4B illustrates the broadcast channel of Figure 4A with an added computer. Computer K was connected to the broadcast channel by edge pinning edges E-J and B-C to computer K. The diameter of this broadcast channel is three, because the shortest path from computer G to computer K is through edges G-A, A-E, and E-K. Figure 4C also illustrates the broadcast channel of Figure 4A with an added computer. Computer K was connected to the broadcast channel by edge pinning edges D-G and E-J to computer K. The diameter of this broadcast channel is, however, still two. Thus, the selection of neighbors impacts the diameter of the broadcast channel. To help minimize the diameter, the broadcast technique uses a random selection technique to identify the four neighbors of a computer in the seeking connection state. The random selection technique tends to distribute the connections to new seeking computers throughout the computers of the broadcast channel which may result in smaller overall diameters.

Broadcasting Through the Graph

As described above, each computer that is connected to the broadcast channel can broadcast messages onto the broadcast channel and does receive all messages that are broadcast on the broadcast channel. The computer that originates a message to be broadcast sends that message to each of its four neighbors using the internal connections. When a computer receives a broadcast message from a neighbor, it sends the message to its three other neighbors. Each computer on the broadcast channel, except the originating computer, will thus receive a copy of each broadcast message from each of its four neighbors. Each computer, however, only sends the first copy of the message that it receives to its neighbors and disregards subsequently received copies. Thus, the total number of copies of a message that is sent between the computers is $3N+1$, where N is the number of computers connected to the broadcast channel. Each computer sends three copies of the message, except for the originating computer, which sends four copies of the message.

The redundancy of the message sending helps to ensure the overall reliability of the broadcast channel. Since each computer has four connections to the broadcast channel, if one computer fails during the broadcast of a message, its neighbors have three other connections through which they will receive copies of the broadcast message. Also, if the internal connection between two computers is slow, each computer has three other connections through which it may receive a copy of each message sooner.

Each computer that originates a message numbers its own messages sequentially. Because of the dynamic nature of the broadcast channel and because there are many possible connection paths between computers, the messages may be received out of order. For example, the distance between an originating computer and a certain receiving computer may be four. After sending the first message, the originating computer and receiving computer may become neighbors and thus the distance between them changes to one. The first message may have to travel a distance of four to reach the receiving computer. The second message only has to travel a distance of one. Thus, it is possible for the second message to reach the receiving computer before the first message.

When the broadcast channel is in a steady state (*i.e.*, no computers connecting or disconnecting from the broadcast channel), out-of-order messages are not a problem because each computer will eventually receive both messages and can queue messages until all earlier ordered messages are received. If, however, the broadcast channel is not in a

steady state, then problems can occur. In particular, a computer may connect to the broadcast channel after the second message has already been received and forwarded on by its new neighbors. When a new neighbor eventually receives the first message, it sends the message to the newly connected computer. Thus, the newly connected computer will receive the first message, but will not receive the second message. If the newly connected computer needs to process the messages in order, it would wait indefinitely for the second message.

One solution to this problem is to have each computer queue all the messages that it receives until it can send them in their proper order to its neighbors. This solution, however, may tend to slow down the propagation of messages through the computers of the broadcast channel. Another solution that may have less impact on the propagation speed is to queue messages only at computers who are neighbors of the newly connected computers. Each already connected neighbor would forward messages as it receives them to its other neighbors who are not newly connected, but not to the newly connected neighbor. The already connected neighbor would only forward messages from each originating computer to the newly connected computer when it can ensure that no gaps in the messages from that originating computer will occur. In one embodiment, the already connected neighbor may track the highest sequence number of the messages already received and forwarded on from each originating computer. The already connected computer will send only higher numbered messages from the originating computers to the newly connected computer. Once all lower numbered messages have been received from all originating computers, then the already connected computer can treat the newly connected computer as its other neighbors and simply forward each message as it is received. In another embodiment, each computer may queue messages and only forwards to the newly connected computer those messages as the gaps are filled in. For example, a computer might receive messages 4 and 5 and then receive message 3. In such a case, the already connected computer would forward queue messages 4 and 5. When message 3 is finally received, the already connected computer will send messages 3, 4, and 5 to the newly connected computer. If messages 4 and 5 were sent to the newly connected computer before message 3, then the newly connected computer would process messages 4 and 5 and disregard message 3. Because the already connected computer queues messages 4 and 5, the newly connected computer will be able to process message 3. It is possible that a newly connected computer will receive a set of messages from an originating computer through one neighbor and then receive another set of message from the

same originating computer through another neighbor. If the second set of messages contains a message that is ordered earlier than the messages of the first set received, then the newly connected computer may ignore that earlier ordered message if the computer already processed those later ordered messages.

5 Decomposing the Graph

A connected computer disconnects from the broadcast channel either in a planned or unplanned manner. When a computer disconnects in a planned manner, it sends a disconnect message to each of its four neighbors. The disconnect message includes a list that identifies the four neighbors of the disconnecting computer. When a neighbor receives the disconnect message, it tries to connect to one of the computers on the list. In one embodiment, the first computer in the list will try to connect to the second computer in the list, and the third computer in the list will try to connect to the fourth computer in the list. If a computer cannot connect (*e.g.*, the first and second computers are already connected), then the computers may try connecting in various other combinations. If connections cannot be established, each computer broadcasts a message that it needs to establish a connection with another computer. When a computer with an available internal port receives the message, it can then establish a connection with the computer that broadcast the message. Figures 5A-5D illustrate the disconnecting of a computer from the broadcast channel. Figure 5A illustrates the disconnecting of a computer from the broadcast channel in a planned manner. When computer H decides to disconnect, it sends its list of neighbors to each of its neighbors (computers A, E, F and I) and then disconnects from each of its neighbors. When computers A and I receive the message they establish a connection between them as indicated by the dashed line, and similarly for computers E and F.

When a computer disconnects in an unplanned manner, such as resulting from a power failure, the neighbors connected to the disconnected computer recognize the disconnection when each attempts to send its next message to the now disconnected computer. Each former neighbor of the disconnected computer recognizes that it is short one connection (*i.e.*, it has a hole or empty port). When a connected computer detects that one of its neighbors is now disconnected, it broadcasts a port connection request on the broadcast channel, which indicates that it has one internal port that needs a connection. The port connection request identifies the call-in port of the requesting computer. When a connected

computer that is also short a connection receives the connection request, it communicates with the requesting computer through its external port to establish a connection between the two computers. Figure 5B illustrates the disconnecting of a computer from the broadcast channel in an unplanned manner. In this illustration, computer H has disconnected in an unplanned manner. When each of its neighbors, computers A, E, F, and I, recognizes the disconnection, each neighbor broadcasts a port connection request indicating that it needs to fill an empty port. As shown by the dashed lines, computers F and I and computers A and E respond to each other's requests and establish a connection.

It is possible that a planned or unplanned disconnection may result in two neighbors each having an empty internal port. In such a case, since they are neighbors, they are already connected and cannot fill their empty ports by connecting to each other. Such a condition is referred to as the "neighbors with empty ports" condition. Each neighbor broadcasts a port connection request when it detects that it has an empty port as described above. When a neighbor receives the port connection request from the other neighbor, it will recognize the condition that its neighbor also has an empty port. Such a condition may also occur when the broadcast channel is in the small regime. The condition can only be corrected when in the large regime. When in the small regime, each computer will have less than four neighbors. To detect this condition in the large regime, which would be a problem if not repaired, the first neighbor to receive the port connection request recognizes the condition and sends a condition check message to the other neighbor. The condition check message includes a list of the neighbors of the sending computer. When the receiving computer receives the list, it compares the list to its own list of neighbors. If the lists are different, then this condition has occurred in the large regime and repair is needed. To repair this condition, the receiving computer will send a condition repair request to one of the neighbors of the sending computer which is not already a neighbor of the receiving computer. When the computer receives the condition repair request, it disconnects from one of its neighbors (other than the neighbor that is involved with the condition) and connects to the computer that sent the condition repair request. Thus, one of the original neighbors involved in the condition will have had a port filled. However, two computers are still in need of a connection, the other original neighbor and the computer that is now disconnected from the computer that received the condition repair request. Those two computers send out port connection requests. If those two computers are not neighbors, then they will connect to

each other when they receive the requests. If, however, the two computers are neighbors, then they repeat the condition repair process until two non-neighbors are in need of connections.

It is possible that the two original neighbors with the condition may have the same set of neighbors. When the neighbor that receives the condition check message determines that the sets of neighbors are the same, it sends a condition double check message to one of its neighbors other than the neighbor who also has the condition. When the computer receives the condition double check message, it determines whether it has the same set of neighbors as the sending computer. If so, the broadcast channel is in the small regime and the condition is not a problem. If the set of neighbors are different, then the computer that received the condition double check message sends a condition check message to the original neighbors with the condition. The computer that receives that condition check message directs one of its neighbors to connect to one of the original neighbors with the condition by sending a condition repair message. Thus, one of the original neighbors with the condition will have its port filled.

Figure 5C illustrates the neighbors with empty ports condition. In this illustration, computer H disconnected in an unplanned manner, but computers F and I responded to the port connection request of the other and are now connected together. The other former neighbors of computer H, computers A and E, are already neighbors, which gives rise to the neighbors with empty ports condition. In this example, computer E received the port connection request from computer A, recognized the possible condition, and sent (since they are neighbors via the internal connection) a condition check message with a list of its neighbors to computer A. When computer A received the list, it recognized that computer E has a different set of neighbor (*i.e.*, the broadcast channel is in the large regime). Computer A selected computer D, which is a neighbor of computer E and sent it a condition repair request. When computer D received the condition repair request, it disconnected from one of its neighbors (other than computer E), which is computer G in this example. Computer D then connected to computer A. Figure 5D illustrates two computers that are not neighbors who now have empty ports. Computers E and G now have empty ports and are not currently neighbors. Therefore, computers E and G can connect to each other.

Figures 5E and 5F further illustrate the neighbors with empty ports condition. Figure 5E illustrates the neighbors with empty ports condition in the small regime. In this

example, if computer E disconnected in an unplanned manner, then each computer broadcasts a port connection request when it detects the disconnect. When computer A receives the port connection request from computer B, it detects the neighbors with empty ports condition and sends a condition check message to computer B. Computer B recognizes that it has the same set of neighbors (computer C and D) as computer A and then sends a condition double check message to computer C. Computer C recognizes that the broadcast channel is in the small regime because is also has the same set of neighbors as computers A and B, computer C may then broadcast a message indicating that the broadcast channel is in the small regime.

Figure 5F illustrates the situation of Figure 5E when in the large regime. As discussed above, computer C receives the condition double check message from computer B. In this case, computer C recognizes that the broadcast channel is in the large regime because it has a set of neighbors that is different from computer B. The edges extending up from computer C and D indicate connections to other computers. Computer C then sends a condition check message to computer B. When computer B receives the condition check message, it sends a condition repair message to one of the neighbors of computer C. The computer that receives the condition repair message disconnects from one of its neighbors, other than computer C, and tries to connect to computer B and the neighbor from which it disconnected tries to connect to computer A.

Port Selection

As described above, the TCP/IP protocol designates ports above number 2056 as user ports. The broadcast technique uses five user port numbers on each computer: one external port and four internal ports. Generally, user ports cannot be statically allocated to an application program because other applications programs executing on the same computer may use conflicting port numbers. As a result, in one embodiment, the computers connected to the broadcast channel dynamically allocate their port numbers. Each computer could simply try to locate the lowest number unused port on that computer and use that port as the call-in port. A seeking computer, however, does not know in advance the call-in port number of the portal computers when the port numbers are dynamically allocated. Thus, a seeking computer needs to dial ports of a portal computer starting with the lowest port number when locating the call-in port of a portal computer. If the portal computer is

connected to (or attempting to connect to) the broadcast channel, then the seeking computer would eventually find the call-in port. If the portal computer is not connected, then the seeking computer would eventually dial every user port. In addition, if each application program on a computer tried to allocate low-ordered port numbers, then a portal computer may end up with a high-numbered port for its call-in port because many of the low-ordered port numbers would be used by other application programs. Since the dialing of a port is a relatively slow process, it would take the seeking computer a long time to locate the call-in port of a portal computer. To minimize this time, the broadcast technique uses a port ordering algorithm to identify the port number order that a portal computer should use when finding an available port for its call-in port. In one embodiment, the broadcast technique uses a hashing algorithm to identify the port order. The algorithm preferably distributes the ordering of the port numbers randomly through out the user port number space and only selects each port number once. In addition, every time the algorithm is executed on any computer for a given channel type and channel instance, it generates the same port ordering. As described below, it is possible for a computer to be connected to multiple broadcast channels that are uniquely identified by channel type and channel instance. The algorithm may be "seeded" with channel type and channel instance in order to generate a unique ordering of port numbers for each broadcast channel. Thus, a seeking computer will dial the ports of a portal computer in the same order as the portal computer used when allocating its call-in port.

If many computers are at the same time seeking connection to a broadcast channel through a single portal computer, then the ports of the portal computer may be busy when called by seeking computers. The seeking computers would typically need to keep on redialing a busy port. The process of locating a call-in port may be significantly slowed by such redialing. In one embodiment, each seeking computer may each reorder the first few port numbers generated by the hashing algorithm. For example, each seeking computer could randomly reorder the first eight port numbers generated by the hashing algorithm. The random ordering could also be weighted where the first port number generated by the hashing algorithm would have a 50% chance of being first in the reordering, the second port number would have a 25% chance of being first in the reordering, and so on. Because the seeking computers would use different orderings, the likelihood of finding a busy port is reduced. For example, if the first eight port numbers are randomly selected, then it is

possible that eight seeking computers could be simultaneously dialing ports in different sequences which would reduce the chances of dialing a busy port.

Locating a Portal Computer

Each computer that can connect to the broadcast channel has a list of one or more portal computers through which it can connect to the broadcast channel. In one embodiment, each computer has the same set of portal computers. A seeking computer locates a portal computer that is connected to the broadcast channel by successively dialing the ports of each portal computer in the order specified by an algorithm. A seeking computer could select the first portal computer and then dial all its ports until a call-in port of a computer that is fully connected to the broadcast channel is found. If no call-in port is found, then the seeking computer would select the next portal computer and repeat the process until a portal computer with such a call-in port is found. A problem with such a seeking technique is that all user ports of each portal computer are dialed until a portal computer fully connected to the broadcast channel is found. In an alternate embodiment, the seeking computer selects a port number according to the algorithm and then dials each portal computer at that port number. If no acceptable call-in port to the broadcast channel is found, then the seeking computer selects the next port number and repeats the process. Since the call-in ports are likely allocated at lower-ordered port numbers, the seeking computer first dials the port numbers that are most likely to be call-in ports of the broadcast channel. The seeking computers may have a maximum search depth, that is the number of ports that it will dial when seeking a portal computer that is fully connected. If the seeking computer exhausts its search depth, then either the broadcast channel has not yet been established or, if the seeking computer is also a portal computer, it can then establish the broadcast channel with itself as the first fully connected computer.

When a seeking computer locates a portal computer that is itself not fully connected, the two computers do not connect when they first locate each other because the broadcast channel may already be established and accessible through a higher-ordered port number on another portal computer. If the two seeking computers were to connect to each other, then two disjoint broadcast channels would be formed. Each seeking computer can share its experience in trying to locate a portal computer with the other seeking computer. In particular, if one seeking computer has searched all the portal computers to a depth of eight,

then the one seeking computer can share that it has searched to a depth of eight with another seeking computer. If that other seeking computer has searched to a depth of, for example, only four, it can skip searching through depths five through eight and that other seeking computer can advance its searching to a depth of nine.

5 In one embodiment, each computer may have a different set of portal computers and a different maximum search depth. In such a situation, it may be possible that two disjoint broadcast channels are formed because a seeking computer cannot locate a fully connected port computer at a higher depth. Similarly, if the set of portal computers are disjoint, then two separate broadcast channels would be formed.

10 Identifying Neighbors for a Seeking Computer

As described above, the neighbors of a newly connecting computer are preferably selected randomly from the set of currently connected computers. One advantage of the broadcast channel, however, is that no computer has global knowledge of the broadcast channel. Rather, each computer has local knowledge of itself and its neighbors. This limited local knowledge has the advantage that all the connected computers are peers (as far as the broadcasting is concerned) and the failure of any one computer (actually any three computers when in the 4-regular and 4-connect form) will not cause the broadcast channel to fail. This local knowledge makes it difficult for a portal computer to randomly select four neighbors for a seeking computer.

20 To select the four computers, a portal computer sends an edge connection request message through one of its internal connections that is randomly selected. The receiving computer again sends the edge connection request message through one of its internal connections that is randomly selected. This sending of the message corresponds to a random walk through the graph that represents the broadcast channel. Eventually, a receiving computer will decide that the message has traveled far enough to represent a randomly selected computer. That receiving computer will offer the internal connection upon which it received the edge connection request message to the seeking computer for edge pinning. Of course, if either of the computers at the end of the offered internal connection are already neighbors of the seeking computer, then the seeking computer cannot
25
30 connect through that internal connection. The computer that decided that the message has

traveled far enough will detect this condition of already being a neighbor and send the message to a randomly selected neighbor.

In one embodiment, the distance that the edge connection request message travels is established by the portal computer to be approximately twice the estimated diameter of the broadcast channel. The message includes an indication of the distance that it is to travel. Each receiving computer decrements that distance to travel before sending the message on. The computer that receives a message with a distance to travel that is zero is considered to be the randomly selected computer. If that randomly selected computer cannot connect to the seeking computer (*e.g.*, because it is already connected to it), then that randomly selected computer forwards the edge connection request to one of its neighbors with a new distance to travel. In one embodiment, the forwarding computer toggles the new distance to travel between zero and one to help prevent two computers from sending the message back and forth between each other.

Because of the local nature of the information maintained by each computer connected to the broadcast channel, the computers need not generally be aware of the diameter of the broadcast channel. In one embodiment, each message sent through the broadcast channel has a distance traveled field. Each computer that forwards a message increments the distance traveled field. Each computer also maintains an estimated diameter of the broadcast channel. When a computer receives a message that has traveled a distance that indicates that the estimated diameter is too small, it updates its estimated diameter and broadcasts an estimated diameter message. When a computer receives an estimated diameter message that indicates a diameter that is larger than its own estimated diameter, it updates its own estimated diameter. This estimated diameter is used to establish the distance that an edge connection request message should travel.

External Data Representation

The computers connected to the broadcast channel may internally store their data in different formats. For example, one computer may use 32-bit integers, and another computer may use 64-bit integers. As another example, one computer may use ASCII to represent text and another computer may use Unicode. To allow communications between heterogeneous computers, the messages sent over the broadcast channel may use the XDR ("eXternal Data Representation") format.

The underlying peer-to-peer communications protocol may send multiple messages in a single message stream. The traditional technique for retrieving messages from a stream has been to repeatedly invoke an operating system routine to retrieve the next message in the stream. The retrieval of each message may require two calls to the operating system: one to retrieve the size of the next message and the other to retrieve the number of bytes indicated by the retrieved size. Such calls to the operating system can, however, be very slow in comparison to the invocations of local routines. To overcome the inefficiencies of such repeated calls, the broadcast technique in one embodiment, uses XDR to identify the message boundaries in a stream of messages. The broadcast technique may request the operating system to provide the next, for example, 1,024 bytes from the stream. The broadcast technique can then repeatedly invoke the XDR routines to retrieve the messages and use the success or failure of each invocation to determine whether another block of 1,024 bytes needs to be retrieved from the operating system. The invocation of XDR routines do not involve system calls and are thus more efficient than repeated system calls.

M-Regular

In the embodiment described above, each fully connected computer has four internal connections. The broadcast technique can be used with other numbers of internal connections. For example, each computer could have 6, 8, or any even number of internal connections. As the number of internal connections increase, the diameter of the broadcast channel tends to decrease, and thus propagation time for a message tends to decrease. The time that it takes to connect a seeking computer to the broadcast channel may, however, increase as the number of internal connections increases. When the number of internal connectors is even, then the broadcast channel can be maintained as m-regular and m-connected (in the steady state). If the number of internal connections is odd, then when the broadcast channel has an odd number of computers connected, one of the computers will have less than that odd number of internal connections. In such a situation, the broadcast network is neither m-regular nor m-connected. When the next computer connects to the broadcast channel, it can again become m-regular and m-connected. Thus, with an odd number of internal connections, the broadcast channel toggles between being and not being m-regular and m-connected.

Components

Figure 6 is a block diagram illustrating components of a computer that is connected to a broadcast channel. The above description generally assumed that there was only one broadcast channel and that each computer had only one connection to that broadcast channel. More generally, a network of computers may have multiple broadcast channels, each computer may be connected to more than one broadcast channel, and each computer can have multiple connections to the same broadcast channel. The broadcast channel is well suited for computer processes (*e.g.*, application programs) that execute collaboratively, such as network meeting programs. Each computer process can connect to one or more broadcast channels. The broadcast channels can be identified by channel type (*e.g.*, application program name) and channel instance that represents separate broadcast channels for that channel type. When a process attempts to connect to a broadcast channel, it seeks a process currently connected to that broadcast channel that is executing on a portal computer. The seeking process identifies the broadcast channel by channel type and channel instance.

Computer 600 includes multiple application programs 601 executing as separate processes. Each application program interfaces with a broadcaster component 602 for each broadcast channel to which it is connected. The broadcaster component may be implemented as an object that is instantiated within the process space of the application program. Alternatively, the broadcaster component may execute as a separate process or thread from the application program. In one embodiment, the broadcaster component provides functions (*e.g.*, methods of class) that can be invoked by the application programs. The primary functions provided may include a connect function that an application program invokes passing an indication of the broadcast channel to which the application program wants to connect. The application program may provide a callback routine that the broadcaster component invokes to notify the application program that the connection has been completed, that is the process enters the fully connected state. The broadcaster component may also provide an acquire message function that the application program can invoke to retrieve the next message that is broadcast on the broadcast channel. Alternatively, the application program may provide a callback routine (which may be a virtual function provided by the application program) that the broadcaster component invokes to notify the application program that a broadcast message has been received. Each broadcaster component allocates a call-in port using the hashing algorithm. When calls are answered at

the call-in port, they are transferred to other ports that serve as the external and internal ports.

The computers connecting to the broadcast channel may include a central processing unit, memory, input devices (e.g., keyboard and pointing device), output devices (e.g., display devices), and storage devices (e.g., disk drives). The memory and storage devices are computer-readable medium that may contain computer instructions that implement the broadcaster component. In addition, the data structures and message structures may be stored or transmitted via a signal transmitted on a computer-readable media, such as a communications link.

Figure 7 is a block diagram illustrating the sub-components of the broadcaster component in one embodiment. The broadcaster component includes a connect component 701, an external dispatcher 702, an internal dispatcher 703 for each internal connection, an acquire message component 704 and a broadcast component 712. The application program may provide a connect callback component 710 and a receive response component 711 that are invoked by the broadcaster component. The application program invokes the connect component to establish a connection to a designated broadcast channel. The connect component identifies the external port and installs the external dispatcher for handling messages that are received on the external port. The connect component invokes the seek portal computer component 705 to identify a portal computer that is connected to the broadcast channel and invokes the connect request component 706 to ask the portal computer (if fully connected) to select neighbor processes for the newly connecting process. The external dispatcher receives external messages, identifies the type of message, and invokes the appropriate handling routine 707. The internal dispatcher receives the internal messages, identifies the type of message, and invokes the appropriate handling routine 708. The received broadcast messages are stored in the broadcast message queue 709. The acquire message component is invoked to retrieve messages from the broadcast queue. The broadcast component is invoked by the application program to broadcast messages in the broadcast channel.

The following tables list messages sent by the broadcaster components.

EXTERNAL MESSAGES

Message Type	Description
seeking_connection_call	Indicates that a seeking process would like to know whether the receiving process is fully connected to the broadcast channel
connection_request_call	Indicates that the sending process would like the receiving process to initiate a connection of the sending process to the broadcast channel
edge_proposal_call	Indicates that the sending process is proposing an edge through which the receiving process can connect to the broadcast channel (<i>i.e.</i> , edge pinning)
port_connection_call	Indicates that the sending process is proposing a port through which the receiving process can connect to the broadcast channel
connected_stmt	Indicates that the sending process is connected to the broadcast channel
condition_repair_stmt	Indicates that the receiving process should disconnect from one of its neighbors and connect to one of the processes involved in the neighbors with empty port condition

INTERNAL MESSAGES

Message Type	Description
broadcast_stmt	Indicates a message that is being broadcast through the broadcast channel for the application programs
connection_port_search_stmt	Indicates that the designated process is looking for a port through which it can connect to the broadcast channel
connection_edge_search_call	Indicates that the requesting process is looking for an edge through which it can connect to the broadcast channel
connection_edge_search_resp	Indicates whether the edge between this process and the sending neighbor has been accepted by the requesting party
diameter_estimate_stmt	Indicates an estimated diameter of the broadcast channel
diameter_reset_stmt	Indicates to reset the estimated diameter to indicated diameter
disconnect_stmt	Indicates that the sending neighbor is disconnecting from the broadcast channel
condition_check_stmt	Indicates that neighbors with empty port condition have

	been detected
condition_double_check_stmt	Indicates that the neighbors with empty ports have the same set of neighbors
shutdown_stmt	Indicates that the broadcast channel is being shutdown

Flow Diagrams

Figures 8-34 are flow diagrams illustrating the processing of the broadcaster component in one embodiment. Figure 8 is a flow diagram illustrating the processing of the connect routine in one embodiment. This routine is passed a channel type (*e.g.*, application name) and channel instance (*e.g.*, session identifier), that identifies the broadcast channel to which this process wants to connect. The routine is also passed auxiliary information that includes the list of portal computers and a connection callback routine. When the connection is established, the connection callback routine is invoked to notify the application program. When this process invokes this routine, it is in the seeking connection state. When a portal computer is located that is connected and this routine connects to at least one neighbor, this process enters the partially connected state, and when the process eventually connects to four neighbors, it enters the fully connected state. When in the small regime, a fully connected process may have less than four neighbors. In block 801, the routine opens the call-in port through which the process is to communicate with other processes when establishing external and internal connections. The port is selected as the first available port using the hashing algorithm described above. In block 802, the routine sets the connect time to the current time. The connect time is used to identify the instance of the process that is connected through this external port. One process may connect to a broadcast channel of a certain channel type and channel instance using one call-in port and then disconnects, and another process may then connect to that same broadcast channel using the same call-in port. Before the other process becomes fully connected, another process may try to communicate with it thinking it is the fully connected old process. In such a case, the connect time can be used to identify this situation. In block 803, the routine invokes the seek portal computer routine passing the channel type and channel instance. The seek portal computer routine attempts to locate a portal computer through which this process can connect to the broadcast channel for the passed type and instance. In decision block 804, if the seek portal computer routine is

successful in locating a fully connected process on that portal computer, then the routine continues at block 805, else the routine returns an unsuccessful indication. In decision block 805, if no portal computer other than the portal computer on which the process is executing was located, then this is the first process to fully connect to broadcast channel and the routine continues at block 806, else the routine continues at block 808. In block 806, the routine invokes the achieve connection routine to change the state of this process to fully connected. In block 807, the routine installs the external dispatcher for processing messages received through this process' external port for the passed channel type and channel instance. When a message is received through that external port, the external dispatcher is invoked. The routine then returns. In block 808, the routine installs an external dispatcher. In block 809, the routine invokes the connect request routine to initiate the process of identifying neighbors for the seeking computer. The routine then returns.

Figure 9 is a flow diagram illustrating the processing of the seek portal computer routine in one embodiment. This routine is passed the channel type and channel instance of the broadcast channel to which this process wishes to connect. This routine, for each search depth (e.g., port number), checks the portal computers at that search depth. If a portal computer is located at that search depth with a process that is fully connected to the broadcast channel, then the routine returns an indication of success. In blocks 902-911, the routine loops selecting each search depth until a process is located. In block 902, the routine selects the next search depth using a port number ordering algorithm. In decision block 903, if all the search depths have already been selected during this execution of the loop, that is for the currently selected depth, then the routine returns a failure indication, else the routine continues at block 904. In blocks 904-911, the routine loops selecting each portal computer and determining whether a process of that portal computer is connected to (or attempting to connect to) the broadcast channel with the passed channel type and channel instance. In block 904, the routine selects the next portal computer. In decision block 905, if all the portal computers have already been selected, then the routine loops to block 902 to select the next search depth, else the routine continues at block 906. In block 906, the routine dials the selected portal computer through the port represented by the search depth. In decision block 907, if the dialing was successful, then the routine continues at block 908, else the routine loops to block 904 to select the next portal computer. The dialing will be successful if the dialed port is the call-in port of the broadcast channel of the passed channel type and channel

instance of a process executing on that portal computer. In block 908, the routine invokes a contact process routine, which contacts the answering process of the portal computer through the dialed port and determines whether that process is fully connected to the broadcast channel. In block 909, the routine hangs up on the selected portal computer. In decision block 910, if the answering process is fully connected to the broadcast channel, then the routine returns a success indicator, else the routine continues at block 911. In block 911, the routine invokes the check for external call routine to determine whether an external call has been made to this process as a portal computer and processes that call. The routine then loops to block 904 to select the next portal computer.

Figure 10 is a flow diagram illustrating the processing of the contact process routine in one embodiment. This routine determines whether the process of the selected portal computer that answered the call-in to the selected port is fully connected to the broadcast channel. In block 1001, the routine sends an external message (*i.e.*, `seeking_connection_call`) to the answering process indicating that a seeking process wants to know whether the answering process is fully connected to the broadcast channel. In block 1002, the routine receives the external response message from the answering process. In decision block 1003, if the external response message is successfully received (*i.e.*, `seeking_connection_resp`), then the routine continues at block 1004, else the routine returns. Wherever the broadcast component requests to receive an external message, it sets a time out period. If the external message is not received within that time out period, the broadcaster component checks its own call-in port to see if another process is calling it. In particular, the dialed process may be calling the dialing process, which may result in a deadlock situation. The broadcaster component may repeat the receive request several times. If the expected message is not received, then the broadcaster component handles the error as appropriate. In decision block 1004, if the answering process indicates in its response message that it is fully connected to the broadcast channel, then the routine continues at block 1005, else the routine continues at block 1006. In block 1005, the routine adds the selected portal computer to a list of connected portal computers and then returns. In block 1006, the routine adds the answering process to a list of fellow seeking processes and then returns.

Figure 11 is a flow diagram illustrating the processing of the connect request routine in one embodiment. This routine requests a process of a portal computer that was identified as being fully connected to the broadcast channel to initiate the connection of this

process to the broadcast channel. In decision block 1101, if at least one process of a portal computer was located that is fully connected to the broadcast channel, then the routine continues at block 1103, else the routine continues at block 1102. A process of the portal computer may no longer be in the list if it recently disconnected from the broadcast channel.

5 In one embodiment, a seeking computer may always search its entire search depth and find multiple portal computers through which it can connect to the broadcast channel. In block 1102, the routine restarts the process of connecting to the broadcast channel and returns. In block 1103, the routine dials the process of one of the found portal computers through the call-in port. In decision block 1104, if the dialing is successful, then the routine continues at

10 block 1105, else the routine continues at block 1113. The dialing may be unsuccessful if, for example, the dialed process recently disconnected from the broadcast channel. In block 1105, the routine sends an external message to the dialed process requesting a connection to the broadcast channel (*i.e.*, `connection_request_call`). In block 1106, the routine receives the response message (*i.e.*, `connection_request_resp`). In decision block 1107, if the response message is successfully received, then the routine continues at block 1108, else the routine continues at block 1113. In block 1108, the routine sets the expected number of holes (*i.e.*, empty internal connections) for this process based on the received response. When in the large regime, the expected number of holes is zero. When in the small regime, the expected number of holes varies from one to three. In block 1109, the routine sets the estimated diameter of the broadcast channel based on the received response. In decision block 1111, if the dialed process is ready to connect to this process as indicated by the response message, then the routine continues at block 1112, else the routine continues at block 1113. In block 1112, the routine invokes the add neighbor routine to add the answering process as a neighbor to this process. This adding of the answering process typically occurs when the

25 broadcast channel is in the small regime. When in the large regime, the random walk search for a neighbor is performed. In block 1113, the routine hangs up the external connection with the answering process computer and then returns.

Figure 12 is a flow diagram of the processing of the check for external call routine in one embodiment. This routine is invoked to identify whether a fellow seeking process is attempting to establish a connection to the broadcast channel through this process. In block 1201, the routine attempts to answer a call on the call-in port. In decision block 1202, if the answer is successful, then the routine continues at block 1203, else the routine

30

returns. In block 1203, the routine receives the external message from the external port. In decision block 1204, if the type of the message indicates that a seeking process is calling (*i.e.*, `seeking_connection_call`), then the routine continues at block 1205, else the routine returns. In block 1205, the routine sends an external message (*i.e.*, `seeking_connection_resp`) to the other seeking process indicating that this process is also is seeking a connection. In decision block 1206, if the sending of the external message is successful, then the routine continues at block 1207, else the routine returns. In block 1207, the routine adds the other seeking process to a list of fellow seeking processes and then returns. This list may be used if this process can find no process that is fully connected to the broadcast channel. In which case, this process may check to see if any fellow seeking process were successful in connecting to the broadcast channel. For example, a fellow seeking process may become the first process fully connected to the broadcast channel.

Figure 13 is a flow diagram of the processing of the achieve connection routine in one embodiment. This routine sets the state of this process to fully connected to the broadcast channel and invokes a callback routine to notify the application program that the process is now fully connected to the requested broadcast channel. In block 1301, the routine sets the connection state of this process to fully connected. In block 1302, the routine notifies fellow seeking processes that it is fully connected by sending a connected external message to them (*i.e.*, `connected_stmt`). In block 1303, the routine invokes the connect callback routine to notify the application program and then returns.

Figure 14 is a flow diagram illustrating the processing of the external dispatcher routine in one embodiment. This routine is invoked when the external port receives a message. This routine retrieves the message, identifies the external message type, and invokes the appropriate routine to handle that message. This routine loops processing each message until all the received messages have been handled. In block 1401, the routine answers (*e.g.*, picks up) the external port and retrieves an external message. In decision block 1402, if a message was retrieved, then the routine continues at block 1403, else the routine hangs up on the external port in block 1415 and returns. In decision block 1403, if the message type is for a process seeking a connection (*i.e.*, `seeking_connection_call`), then the routine invokes the handle seeking connection call routine in block 1404, else the routine continues at block 1405. In decision block 1405, if the message type is for a connection request call (*i.e.*, `connection_request_call`), then the routine invokes the handle connection

request call routine in block 1406, else the routine continues at block 1407. In decision block 1407, if the message type is edge proposal call (*i.e.*, `edge_proposal_call`), then the routine invokes the handle edge proposal call routine in block 1408, else the routine continues at block 1409. In decision block 1409, if the message type is port connect call (*i.e.*, `port_connect_call`), then the routine invokes the handle port connection call routine in block 1410, else the routine continues at block 1411. In decision block 1411, if the message type is a connected statement (*i.e.*, `connected_stmt`), the routine invokes the handle connected statement in block 1412, else the routine continues at block 1212. In decision block 1412, if the message type is a condition repair statement (*i.e.*, `condition_repair_stmt`), then the routine invokes the handle condition repair routine in block 1413, else the routine loops to block 1414 to process the next message. After each handling routine is invoked, the routine loops to block 1414. In block 1414, the routine hangs up on the external port and continues at block 1401 to receive the next message.

Figure 15 is a flow diagram illustrating the processing of the handle seeking connection call routine in one embodiment. This routine is invoked when a seeking process is calling to identify a portal computer through which it can connect to the broadcast channel. In decision block 1501, if this process is currently fully connected to the broadcast channel identified in the message, then the routine continues at block 1502, else the routine continues at block 1503. In block 1502, the routine sets a message to indicate that this process is fully connected to the broadcast channel and continues at block 1505. In block 1503, the routine sets a message to indicate that this process is not fully connected. In block 1504, the routine adds the identification of the seeking process to a list of fellow seeking processes. If this process is not fully connected, then it is attempting to connect to the broadcast channel. In block 1505, the routine sends the external message response (*i.e.*, `seeking_connection_resp`) to the seeking process and then returns.

Figure 16 is a flow diagram illustrating processing of the handle connection request call routine in one embodiment. This routine is invoked when the calling process wants this process to initiate the connection of the process to the broadcast channel. This routine either allows the calling process to establish an internal connection with this process (*e.g.*, if in the small regime) or starts the process of identifying a process to which the calling process can connect. In decision block 1601, if this process is currently fully connected to the broadcast channel, then the routine continues at block 1603, else the routine hangs up on

the external port in block 1602 and returns. In block 1603, the routine sets the number of holes that the calling process should expect in the response message. In block 1604, the routine sets the estimated diameter in the response message. In block 1605, the routine indicates whether this process is ready to connect to the calling process. This process is ready to connect when the number of its holes is greater than zero and the calling process is not a neighbor of this process. In block 1606, the routine sends to the calling process an external message that is responsive to the connection request call (*i.e.*, `connection_request_resp`). In block 1607, the routine notes the number of holes that the calling process needs to fill as indicated in the request message. In decision block 1608, if this process is ready to connect to the calling process, then the routine continues at block 1609, else the routine continues at block 1611. In block 1609, the routine invokes the add neighbor routine to add the calling process as a neighbor. In block 1610, the routine decrements the number of holes that the calling process needs to fill and continues at block 1611. In block 1611, the routine hangs up on the external port. In decision block 1612, if this process has no holes or the estimated diameter is greater than one (*i.e.*, in the large regime), then the routine continues at block 1613, else the routine continues at block 1616. In blocks 1613-1615, the routine loops forwarding a request for an edge through which to connect to the calling process to the broadcast channel. One request is forwarded for each pair of holes of the calling process that needs to be filled. In decision block 1613, if the number of holes of the calling process to be filled is greater than or equal to two, then the routine continues at block 1614, else the routine continues at block 1616. In block 1614, the routine invokes the forward connection edge search routine. The invoked routine is passed to an indication of the calling process and the random walk distance. In one embodiment, the distance is twice in the estimated diameter of the broadcast channel. In block 1614, the routine decrements the holes left to fill by two and loops to block 1613. In decision block 1616, if there is still a hole to fill, then the routine continues at block 1617, else the routine returns. In block 1617, the routine invokes the fill hole routine passing the identification of the calling process. The fill hole routine broadcasts a connection port search statement (*i.e.*, `connection_port_search_stmt`) for a hole of a connected process through which the calling process can connect to the broadcast channel. The routine then returns.

Figure 17 is a flow diagram illustrating the processing of the add neighbor routine in one embodiment. This routine adds the process calling on the external port as a

neighbor to this process. In block 1701, the routine identifies the calling process on the external port. In block 1702, the routine sets a flag to indicate that the neighbor has not yet received the broadcast messages from this process. This flag is used to ensure that there are no gaps in the messages initially sent to the new neighbor. The external port becomes the internal port for this connection. In decision block 1703, if this process is in the seeking connection state, then this process is connecting to its first neighbor and the routine continues at block 1704, else the routine continues at block 1705. In block 1704, the routine sets the connection state of this process to partially connected. In block 1705, the routine adds the calling process to the list of neighbors of this process. In block 1706, the routine installs an internal dispatcher for the new neighbor. The internal dispatcher is invoked when a message is received from that new neighbor through the internal port of that new neighbor. In decision block 1707, if this process buffered up messages while not fully connected, then the routine continues at block 1708, else the routine continues at block 1709. In one embodiment, a process that is partially connected may buffer the messages that it receives through an internal connection so that it can send these messages as it connects to new neighbors. In block 1708, the routine sends the buffered messages to the new neighbor through the internal port. In decision block 1709, if the number of holes of this process equals the expected number of holes, then this process is fully connected and the routine continues at block 1710, else the routine continues at block 1711. In block 1710, the routine invokes the achieve connected routine to indicate that this process is fully connected. In decision block 1711, if the number of holes for this process is zero, then the routine continues at block 1712, else the routine returns. In block 1712, the routine deletes any pending edges and then returns. A pending edge is an edge that has been proposed to this process for edge pinning, which in this case is no longer needed.

Figure 18 is a flow diagram illustrating the processing of the forward connection edge search routine in one embodiment. This routine is responsible for passing along a request to connect a requesting process to a randomly selected neighbor of this process through the internal port of the selected neighbor, that is part of the random walk. In decision block 1801, if the forwarding distance remaining is greater than zero, then the routine continues at block 1804, else the routine continues at block 1802. In decision block 1802, if the number of neighbors of this process is greater than one, then the routine continues at block 1804, else this broadcast channel is in the small regime and the routine

continues at block 1803. In decision block 1803, if the requesting process is a neighbor of this process, then the routine returns, else the routine continues at block 1804. In blocks 1804-1807, the routine loops attempting to send a connection edge search call internal message (*i.e.*, `connection_edge_search_call`) to a randomly selected neighbor. In block 1804, the routine randomly selects a neighbor of this process. In decision block 1805, if all the neighbors of this process have already been selected, then the routine cannot forward the message and the routine returns, else the routine continues at block 1806. In block 1806, the routine sends a connection edge search call internal message to the selected neighbor. In decision block 1807, if the sending of the message is successful, then the routine continues at block 1808, else the routine loops to block 1804 to select the next neighbor. When the sending of an internal message is unsuccessful, then the neighbor may have disconnected from the broadcast channel in an unplanned manner. Whenever such a situation is detected by the broadcaster component, it attempts to find another neighbor by invoking the fill holes routine to fill a single hole or the forward connecting edge search routine to fill two holes. In block 1808, the routine notes that the recently sent connection edge search call has not yet been acknowledged and indicates that the edge to this neighbor is reserved if the remaining forwarding distance is less than or equal to one. It is reserved because the selected neighbor may offer this edge to the requesting process for edge pinning. The routine then returns.

Figure 19 is a flow diagram illustrating the processing of the handle edge proposal call routine. This routine is invoked when a message is received from a proposing process that proposes to connect an edge between the proposing process and one of its neighbors to this process for edge pinning. In decision block 1901, if the number of holes of this process minus the number of pending edges is greater than or equal to one, then this process still has holes to be filled and the routine continues at block 1902, else the routine continues at block 1911. In decision block 1902, if the proposing process or its neighbor is a neighbor of this process, then the routine continues at block 1911, else the routine continues at block 1903. In block 1903, the routine indicates that the edge is pending between this process and the proposing process. In decision block 1904, if a proposed neighbor is already pending as a proposed neighbor, then the routine continues at block 1911, else the routine continues at block 1907. In block 1907, the routine sends an edge proposal response as an external message to the proposing process (*i.e.*, `edge_proposal_resp`) indicating that the proposed edge is accepted. In decision block 1908, if the sending of the message was

successful, then the routine continues at block 1909, else the routine returns. In block 1909, the routine adds the edge as a pending edge. In block 1910, the routine invokes the add neighbor routine to add the proposing process on the external port as a neighbor. The routine then returns. In block 1911, the routine sends an external message (*i.e.*, `edge_proposal_resp`) indicating that this proposed edge is not accepted. In decision block 1912, if the number of holes is odd, then the routine continues at block 1913, else the routine returns. In block 1913, the routine invokes the fill hole routine and then returns.

Figure 20 is a flow diagram illustrating the processing of the handle port connection call routine in one embodiment. This routine is invoked when an external message is received then indicates that the sending process wants to connect to one hole of this process. In decision block 2001, if the number of holes of this process is greater than zero, then the routine continues at block 2002, else the routine continues at block 2003. In decision block 2002, if the sending process is not a neighbor, then the routine continues at block 2004, else the routine continues to block 2003. In block 2003, the routine sends a port connection response external message (*i.e.*, `port_connection_resp`) to the sending process that indicates that it is not okay to connect to this process. The routine then returns. In block 2004, the routine sends a port connection response external message to the sending process that indicates that is okay to connect this process. In decision block 2005, if the sending of the message was successful, then the routine continues at block 2006, else the routine continues at block 2007. In block 2006, the routine invokes the add neighbor routine to add the sending process as a neighbor of this process and then returns. In block 2007, the routine hangs up the external connection. In block 2008, the routine invokes the connect request routine to request that a process connect to one of the holes of this process. The routine then returns.

Figure 21 is a flow diagram illustrating the processing of the fill hole routine in one embodiment. This routine is passed an indication of the requesting process. If this process is requesting to fill a hole, then this routine sends an internal message to other processes. If another process is requesting to fill a hole, then this routine invokes the routine to handle a connection port search request. In block 2101, the routine initializes a connection port search statement internal message (*i.e.*, `connection_port_search_stmt`). In decision block 2102, if this process is the requesting process, then the routine continues at block 2103, else the routine continues at block 2104. In block 2103, the routine distributes

the message to the neighbors of this process through the internal ports and then returns. In block 2104, the routine invokes the handle connection port search routine and then returns.

Figure 22 is a flow diagram illustrating the processing of the internal dispatcher routine in one embodiment. This routine is passed an indication of the neighbor who sent the internal message. In block 2201, the routine receives the internal message. This routine identifies the message type and invokes the appropriate routine to handle the message. In block 2202, the routine assesses whether to change the estimated diameter of the broadcast channel based on the information in the received message. In decision block 2203, if this process is the originating process of the message or the message has already been received (*i.e.*, a duplicate), then the routine ignores the message and continues at block 2208, else the routine continues at block 2203A. In decision block 2203A, if the process is partially connected, then the routine continues at block 2203B, else the routine continues at block 2204. In block 2203B, the routine adds the message to the pending connection buffer and continues at block 2204. In decision blocks 2204-2207, the routine decodes the message type and invokes the appropriate routine to handle the message. For example, in decision block 2204, if the type of the message is broadcast statement (*i.e.*, broadcast_stmt), then the routine invokes the handle broadcast message routine in block 2205. After invoking the appropriate handling routine, the routine continues at block 2208. In decision block 2208, if the partially connected buffer is full, then the routine continues at block 2209, else the routine continues at block 2210. The broadcaster component collects all its internal messages in a buffer while partially connected so that it can forward the messages as it connects to new neighbors. If, however, that buffer becomes full, then the process assumes that it is now fully connected and that the expected number of connections was too high, because the broadcast channel is now in the small regime. In block 2209, the routine invokes the achieve connection routine and then continues in block 2210. In decision block 2210, if the application program message queue is empty, then the routine returns, else the routine continues at block 2212. In block 2212, the routine invokes the receive response routine passing the acquired message and then returns. The received response routine is a callback routine of the application program.

Figure 23 is a flow diagram illustrating the processing of the handle broadcast message routine in one embodiment. This routine is passed an indication of the originating process, an indication of the neighbor who sent the broadcast message, and the broadcast

message itself. In block 2301, the routine performs the out of order processing for this message. The broadcaster component queues messages from each originating process until it can send them in sequence number order to the application program. In block 2302, the routine invokes the distribute broadcast message routine to forward the message to the neighbors of this process. In decision block 2303, if a newly connected neighbor is waiting to receive messages, then the routine continues at block 2304, else the routine returns. In block 2304, the routine sends the messages in the correct order if possible for each originating process and then returns.

Figure 24 is a flow diagram illustrating the processing of the distribute broadcast message routine in one embodiment. This routine sends the broadcast message to each of the neighbors of this process, except for the neighbor who sent the message to this process. In block 2401, the routine selects the next neighbor other than the neighbor who sent the message. In decision block 2402, if all such neighbors have already been selected, then the routine returns. In block 2403, the routine sends the message to the selected neighbor and then loops to block 2401 to select the next neighbor.

Figure 26 is a flow diagram illustrating the processing of the handle connection port search statement routine in one embodiment. This routine is passed an indication of the neighbor that sent the message and the message itself. In block 2601, the routine invokes the distribute internal message which sends the message to each of its neighbors other than the sending neighbor. In decision block 2602, if the number of holes of this process is greater than zero, then the routine continues at block 2603, else the routine returns. In decision block 2603, if the requesting process is a neighbor, then the routine continues at block 2605, else the routine continues at block 2604. In block 2604, the routine invokes the court neighbor routine and then returns. The court neighbor routine connects this process to the requesting process if possible. In block 2605, if this process has one hole, then the neighbors with empty ports condition exists and the routine continues at block 2606, else the routine returns. In block 2606, the routine generates a condition check message (*i.e.*, `condition_check`) that includes a list of this process' neighbors. In block 2607, the routine sends the message to the requesting neighbor.

Figure 27 is a flow diagram illustrating the processing of the court neighbor routine in one embodiment. This routine is passed an indication of the prospective neighbor for this process. If this process can connect to the prospective neighbor, then it sends a port

connection call external message to the prospective neighbor and adds the prospective neighbor as a neighbor. In decision block 2701, if the prospective neighbor is already a neighbor, then the routine returns, else the routine continues at block 2702. In block 2702, the routine dials the prospective neighbor. In decision block 2703, if the number of holes of this process is greater than zero, then the routine continues at block 2704, else the routine continues at block 2706. In block 2704, the routine sends a port connection call external message (*i.e.*, port_connection_call) to the prospective neighbor and receives its response (*i.e.*, port_connection_resp). Assuming the response is successfully received, in block 2705, the routine adds the prospective neighbor as a neighbor of this process by invoking the add neighbor routine. In block 2706, the routine hangs up with the prospect and then returns.

Figure 28 is a flow diagram illustrating the processing of the handle connection edge search call routine in one embodiment. This routine is passed a indication of the neighbor who sent the message and the message itself. This routine either forwards the message to a neighbor or proposes the edge between this process and the sending neighbor to the requesting process for edge pinning. In decision block 2801, if this process is not the requesting process or the number of holes of the requesting process is still greater than or equal to two, then the routine continues at block 2802, else the routine continues at block 2813. In decision block 2802, if the forwarding distance is greater than zero, then the random walk is not complete and the routine continues at block 2803, else the routine continues at block 2804. In block 2803, the routine invokes the forward connection edge search routine passing the identification of the requesting process and the decremented forwarding distance. The routine then continues at block 2815. In decision block 2804, if the requesting process is a neighbor or the edge between this process and the sending neighbor is reserved because it has already been offered to a process, then the routine continues at block 2805, else the routine continues at block 2806. In block 2805, the routine invokes the forward connection edge search routine passing an indication of the requesting party and a toggle indicator that alternatively indicates to continue the random walk for one or two more computers. The routine then continues at block 2815. In block 2806, the routine dials the requesting process via the call-in port. In block 2807, the routine sends an edge proposal call external message (*i.e.*, edge_proposal_call) and receives the response (*i.e.*, edge_proposal_resp). Assuming that the response is successfully received, the routine continues at block 2808. In decision block 2808, if the response indicates that the edge is

acceptable to the requesting process, then the routine continues at block 2809, else the routine continues at block 2812. In block 2809, the routine reserves the edge between this process and the sending neighbor. In block 2810, the routine adds the requesting process as a neighbor by invoking the add neighbor routine. In block 2811, the routine removes the sending neighbor as a neighbor. In block 2812, the routine hangs up the external port and continues at block 2815. In decision block 2813, if this process is the requesting process and the number of holes of this process equals one, then the routine continues at block 2814, else the routine continues at block 2815. In block 2814, the routine invokes the fill hole routine. In block 2815, the routine sends an connection edge search response message (*i.e.*, connection_edge_search_response) to the sending neighbor indicating acknowledgement and then returns. The graphs are sensitive to parity. That is, all possible paths starting from a node and ending at that node will have an even length unless the graph has a cycle whose length is odd. The broadcaster component uses a toggle indicator to vary the random walk distance between even and odd distances.

Figure 29 is a flow diagram illustrating the processing of the handle connection edge search response routine in one embodiment. This routine is passed as indication of the requesting process, the sending neighbor, and the message. In block 2901, the routine notes that the connection edge search response (*i.e.*, connection_edge_search_resp) has been received and if the forwarding distance is less than or equal to one unreserves the edge between this process and the sending neighbor. In decision block 2902, if the requesting process indicates that the edge is acceptable as indicated in the message, then the routine continues at block 2903, else the routine returns. In block 2903, the routine reserves the edge between this process and the sending neighbor. In block 2904, the routine removes the sending neighbor as a neighbor. In block 2905, the routine invokes the court neighbor routine to connect to the requesting process. In decision block 2906, if the invoked routine was unsuccessful, then the routine continues at block 2907, else the routine returns. In decision block 2907, if the number of holes of this process is greater than zero, then the routine continues at block 2908, else the routine returns. In block 2908, the routine invokes the fill hole routine and then returns.

Figure 30 is a flow diagram illustrating the processing of the broadcast routine in one embodiment. This routine is invoked by the application program to broadcast a message on the broadcast channel. This routine is passed the message to be broadcast. In

5 decision block 3001, if this process has at least one neighbor, then the routine continues at block 3002, else the routine returns since it is the only process connected to be broadcast channel. In block 3002, the routine generates an internal message of the broadcast statement type (*i.e.*, broadcast _stmt). In block 3003, the routine sets the sequence number of the message. In block 3004, the routine invokes the distribute internal message routine to broadcast the message on the broadcast channel. The routine returns.

10 Figure 31 is a flow diagram illustrating the processing of the acquire message routine in one embodiment. The acquire message routine may be invoked by the application program or by a callback routine provided by the application program. This routine returns a message. In block 3101, the routine pops the message from the message queue of the broadcast channel. In decision block 3102, if a message was retrieved, then the routine returns an indication of success, else the routine returns indication of failure.

15
20
25 Figures 32-34 are flow diagrams illustrating the processing of messages associated with the neighbors with empty ports condition. Figure 32 is a flow diagram illustrating processing of the handle condition check message in one embodiment. This message is sent by a neighbor process that has one hole and has received a request to connect to a hole of this process. In decision block 3201, if the number of holes of this process is equal to one, then the routine continues at block 3202, else the neighbors with empty ports condition does not exist any more and the routine returns. In decision block 3202, if the sending neighbor and this process have the same set of neighbors, the routine continues at block 3203, else the routine continues at block 3205. In block 3203, the routine initializes a condition double check message (*i.e.*, condition_double_check) with the list of neighbors of this process. In block 3204, the routine sends the message internally to a neighbor other than sending neighbor. The routine then returns. In block 3205, the routine selects a neighbor of the sending process that is not also a neighbor of this process. In block 3206, the routine sends a condition repair message (*i.e.*, condition_repair_stmt) externally to the selected process. In block 3207, the routine invokes the add neighbor routine to add the selected neighbor as a neighbor of this process and then returns.

30 Figure 33 is a flow diagram illustrating processing of the handle condition repair statement routine in one embodiment. This routine removes an existing neighbor and connects to the process that sent the message. In decision block 3301, if this process has no holes, then the routine continues at block 3302, else the routine continues at block 3304. In

block 3302, the routine selects a neighbor that is not involved in the neighbors with empty ports condition. In block 3303, the routine removes the selected neighbor as a neighbor of this process. Thus, this process that is executing the routine now has at least one hole. In block 3304, the routine invokes the add neighbor routine to add the process that sent the message as a neighbor of this process. The routine then returns.

Figure 34 is a flow diagram illustrating the processing of the handle condition double check routine. This routine determines whether the neighbors with empty ports condition really is a problem or whether the broadcast channel is in the small regime. In decision block 3401, if this process has one hole, then the routine continues at block 3402, else the routine continues at block 3403. If this process does not have one hole, then the set of neighbors of this process is not the same as the set of neighbors of the sending process. In decision block 3402, if this process and the sending process have the same set of neighbors, then the broadcast channel is not in the small regime and the routine continues at block 3403, else the routine continues at block 3406. In decision block 3403, if this process has no holes, then the routine returns, else the routine continues at block 3404. In block 3404, the routine sets the estimated diameter for this process to one. In block 3405, the routine broadcasts a diameter reset internal message (*i.e.*, `diameter_reset`) indicating that the estimated diameter is one and then returns. In block 3406, the routine creates a list of neighbors of this process. In block 3407, the routine sends the condition check message (*i.e.*, `condition_check_stmt`) with the list of neighbors to the neighbor who sent the condition double check message and then returns.

From the above description, it will be appreciated that although specific embodiments of the technology have been described, various modifications may be made without deviating from the spirit and scope of the invention. For example, the communications on the broadcast channel may be encrypted. Also, the channel instance or session identifier may be a very large number (*e.g.*, 128 bits) to help prevent an unauthorized user to maliciously tap into a broadcast channel. The portal computer may also enforce security and not allow an unauthorized user to connect to the broadcast channel. Accordingly, the invention is not limited except by the claims.

CLAIMS

sub
A4

1 1. A computer network having a plurality of participants, each participant
2 having connections to at least three neighbor participants, wherein an originating participant
3 sends data to the other participants by sending the data through each of its connections to its
4 neighbor participants and wherein each participant sends data that it receives from a neighbor
5 participant to its other neighbor participants.

1 2. The computer network of claim 1 wherein each participant is connected
2 to 4 other participants.

1 3. The computer network of claim 1 wherein each participant is connected
2 to an even number of other participants.

1 4. The computer network of claim 1 wherein the network is m-regular,
2 where m is the number of neighbor participants of each participant.

1 5. The computer network of claim 4 wherein the network is m-connected,
2 where m is the number of neighbor participants of each participant.

1 6. The computer network of claim 1 wherein the network is m-regular and
2 m-connected, where m is the number of neighbor participants of each participant.

1 7. The computer network of claim 1 wherein all the participants are peers.

1 8. The computer network of claim 1 wherein the connections are peer-to-
2 peer connections.

1 9. The computer network of claim 1 wherein the connections are TCP/IP
2 connections.

1 10. The computer network of claim 1 wherein each participant is a process
2 executing on a computer.

1 11. The computer network of claim 1 wherein a computer hosts more than
2 one participant.

1 12. The computer network of claim 1 wherein each participant sends to each
2 of its neighbors only one copy of the data.

1 13. A component for controlling communications of a participant with a
2 broadcast channel, comprising:
3 a contact module that locates a portal computer and requests the located
4 portal computer to provide an indication of neighbor participants to which the participant can
5 be connected; and
6 a join module that receives the indication of neighbor participants and
7 establishes a connection between the participant and each of the indicated neighbor
8 participants.

1 14. The component of claim 13 wherein each participant is a computer
2 process.

1 15. The component of claim 13 wherein the indicated participants are
2 computer processes executing on different computer systems.

1 16. The component of claim 13 including:
2 a broadcast module that receives data from a neighbor participant of the
3 participant and transmits the received data to the other neighbor participants.

1 17. The component of claim 13 including:
2 a connection request module that receives a request to connect to
3 another participant, disconnects from a neighbor participant, and connects to the other
4 participant.

1 18. The component of claim 13 wherein the connections are established
2 using the TCP/IP protocol.

1 19. A broadcast channel for participants, comprising:
2 a communications network that provides peer-to-peer communications
3 between the participants connected to the broadcast channel; and
4 for each participant connected to the broadcast channel,
5 an indication of four neighbor participants of that
6 participant; and
7 a broadcast component that receives data from a neighbor
8 participant using the communications network and that sends the received data to its other
9 neighbor participants to effect the broadcasting of the data to each participant of the
10 broadcast channel.

1 20. The broadcast channel of claim 19 wherein the broadcast component
2 disregards received data that it has already sent to its neighbor participants.

1 21. The broadcast channel of claim 19 wherein a participant connects to the
2 broadcast channel by contacting a participant already connected to the broadcast channel.

1 22. The broadcast channel of claim 19 wherein each participant is a
2 computer process.

1 23. The broadcast channel of claim 19 wherein each participant is a
2 computer thread.

1 24. The broadcast channel of claim 19 wherein each participant is a
2 computer.

1 25. The broadcast channel of claim 19 wherein the communications network
2 uses TCP/IP protocol.

1 26. The broadcast channel of claim 19 wherein the communications network
2 is the Internet.

1 27. The broadcast channel of claim 19 wherein the participants are peers.

1 28. A broadcast channel comprising a plurality of participants, each
2 participant being connected to neighbor participants, the participants and connections
3 between them forming an m-regular graph, where m is greater than 2 and the number of
4 participants is greater than m.

1 29. The broadcast channel of claim 28 wherein the graph is m-connected.

1 30. The broadcast channel of claim 28 wherein m is even.

1 31. The broadcast channel of claim 28 wherein m is odd and the number of
2 participants is even.

1 32. The broadcast channel of claim 28 wherein the participants are
2 computer processes.

1 33. The broadcast channel of claim 28 wherein the participants are
2 computers.

1 34. The broadcast channel of claim 28 wherein the connections are
2 established using TCP/IP protocol.

1 35. The broadcast channel of claim 28 wherein a message is broadcast on
2 the broadcast channel by an originating participant sending the message to each of its
3 neighbor participants and by each participant upon receiving a message from a neighbor
4 participant sending the message to its other neighbor participants.

1 36. A broadcast channel comprising a plurality of participants, each
2 participant being connected to neighbor participants, the participants and connections
3 between them form an m-regular graph, where m is greater than 2, and wherein when a
4 participant receives data from a neighbor participant, it sends the data to its other neighbor
5 participants.

114

1 37. The broadcast channel of claim 36 wherein the number of participants is
2 greater than m.

1 38. The broadcast channel of claim 36 wherein the graph is m-connected.

1 39. The broadcast channel of claim 36 wherein m is even.

1 40. The broadcast channel of claim 36 wherein m is odd and the number of
2 participants is even.

1 41. The broadcast channel of claim 36 wherein the participants are
2 computer processes.

1 42. The broadcast channel of claim 36 wherein the participants are
2 computers.

1 43. The broadcast channel of claim 36 wherein the connections are
2 established using TCP/IP protocol.

1 44. A computer-readable medium containing instructions for controlling
2 communications of a participant of a broadcast channel, by a method comprising:
3 locating a portal computer;
4 requesting the located portal computer to provide an indication of
5 neighbor participants to which the participant can be connected;
6 receiving the indications of the neighbor participants; and
7 establishing a connection between the participant and each of the
8 indicated neighbor participants.

1 45. The computer-readable medium of claim 44 wherein each participant is
2 a computer process.

119

3 46. The computer-readable medium of claim 44 wherein the indicated
4 participants are computer processes executing on different computer systems.

5 47. The computer-readable medium of claim 44 including:
6 receiving data from a neighbor participant of the participant; and
7 transmitting the received data to the other neighbor participants.

8 48. The computer-readable medium of claim 44 including:
1 receiving a request to connect to another participant;
2 disconnecting from a neighbor participant; and
3 connecting to the other participant.

4 49. The computer-readable medium of claim 44 wherein the connections are
1 established using the TCP/IP protocol.
2

add
A1

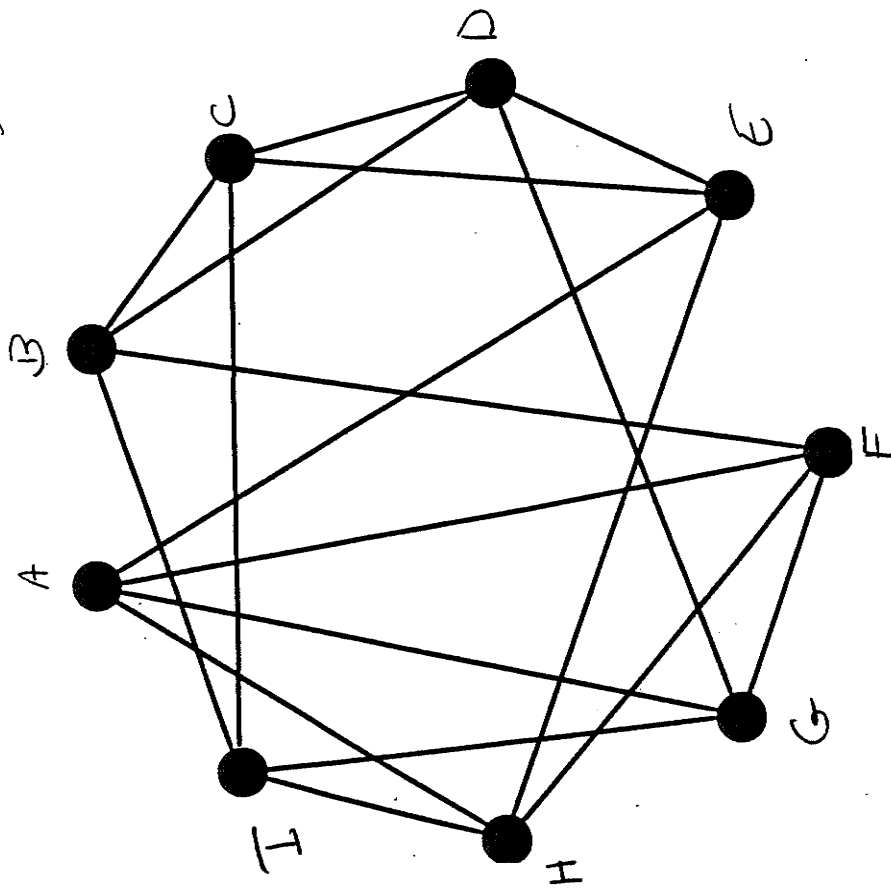


Fig 1

001642 92553960

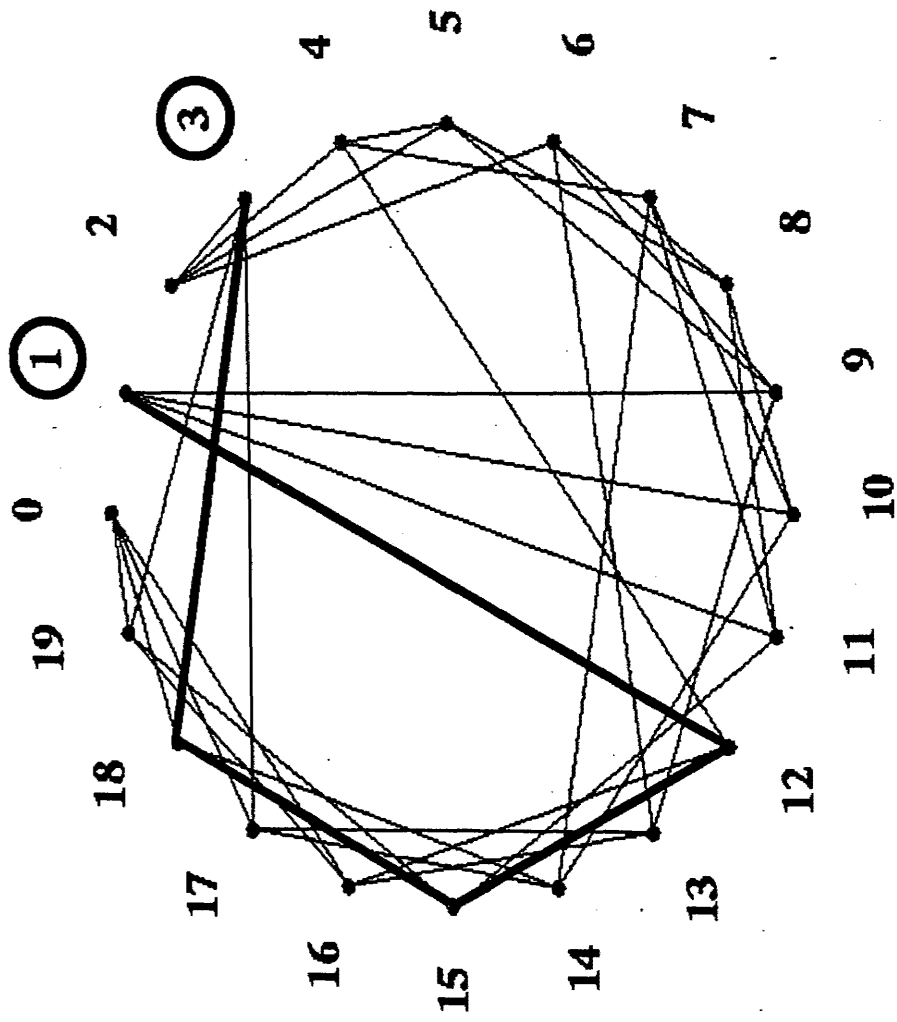


Fig 2

100 100 100 100 100 100 100 100 100 100

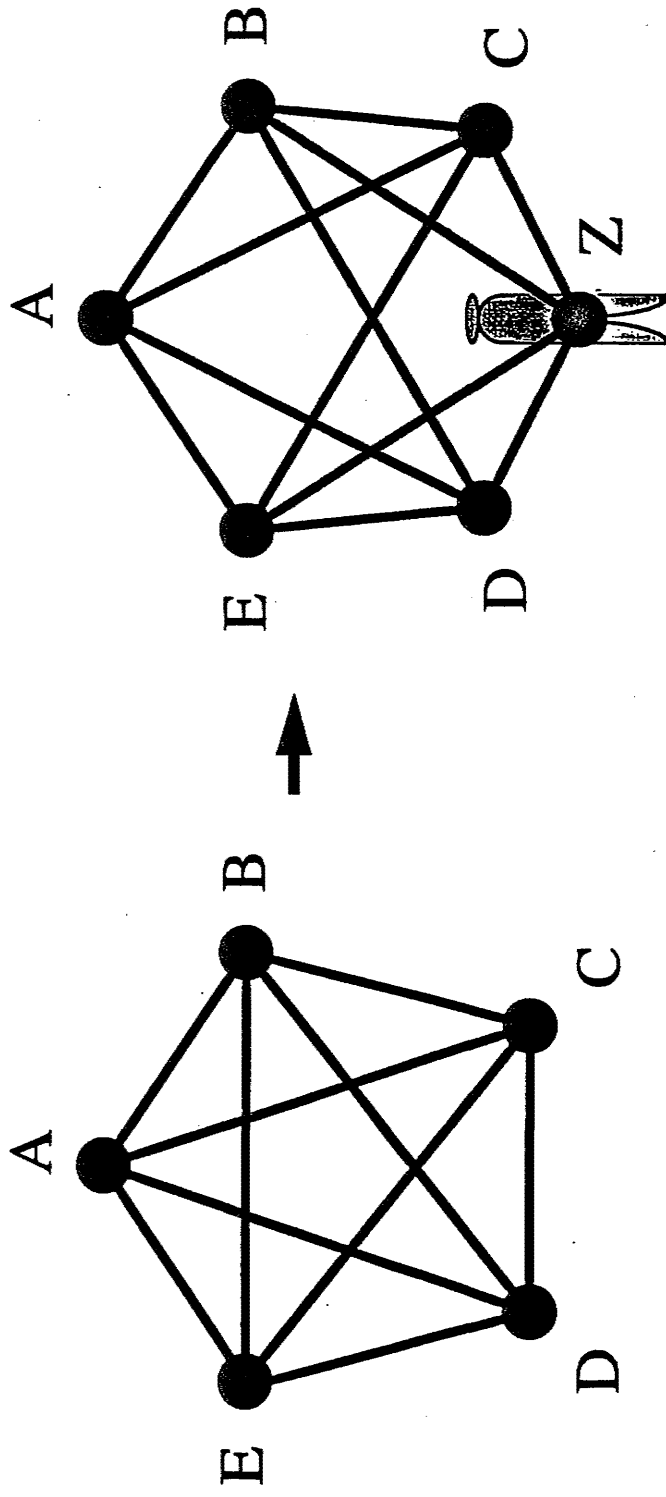


Fig 3B

Fig 3A

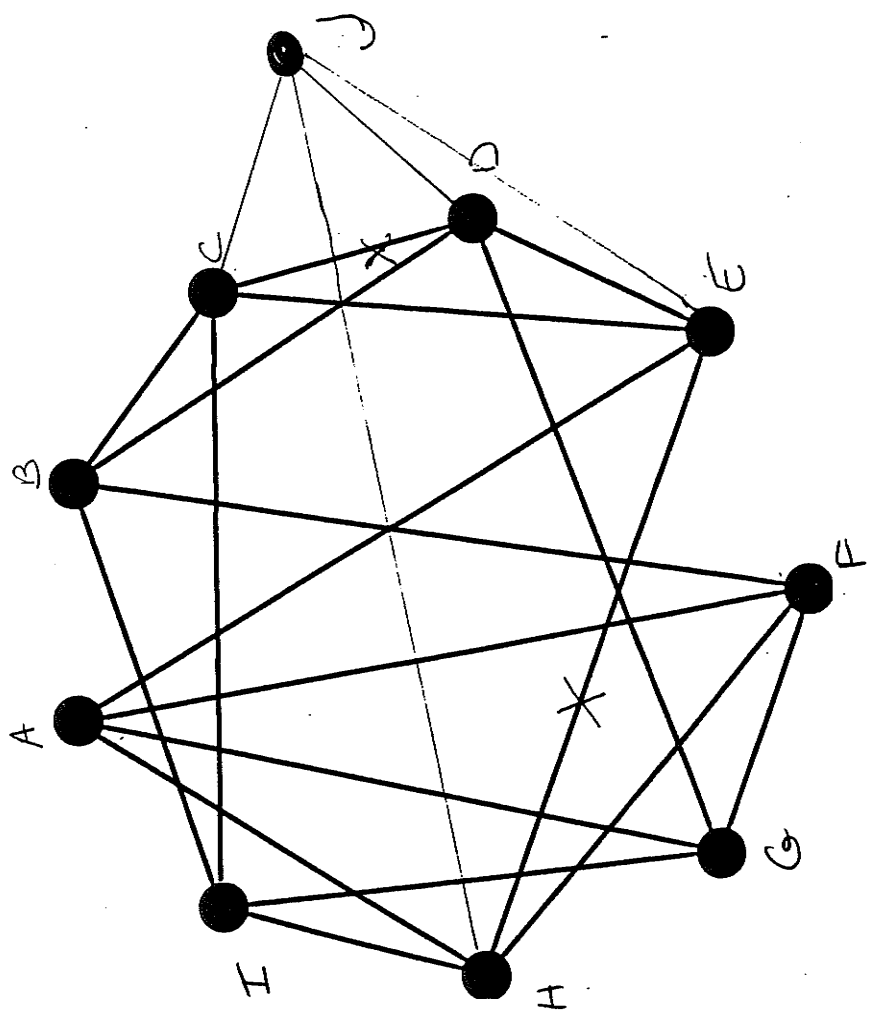


FIG 4A

001 002 003 004 005 006 007 008 009 010

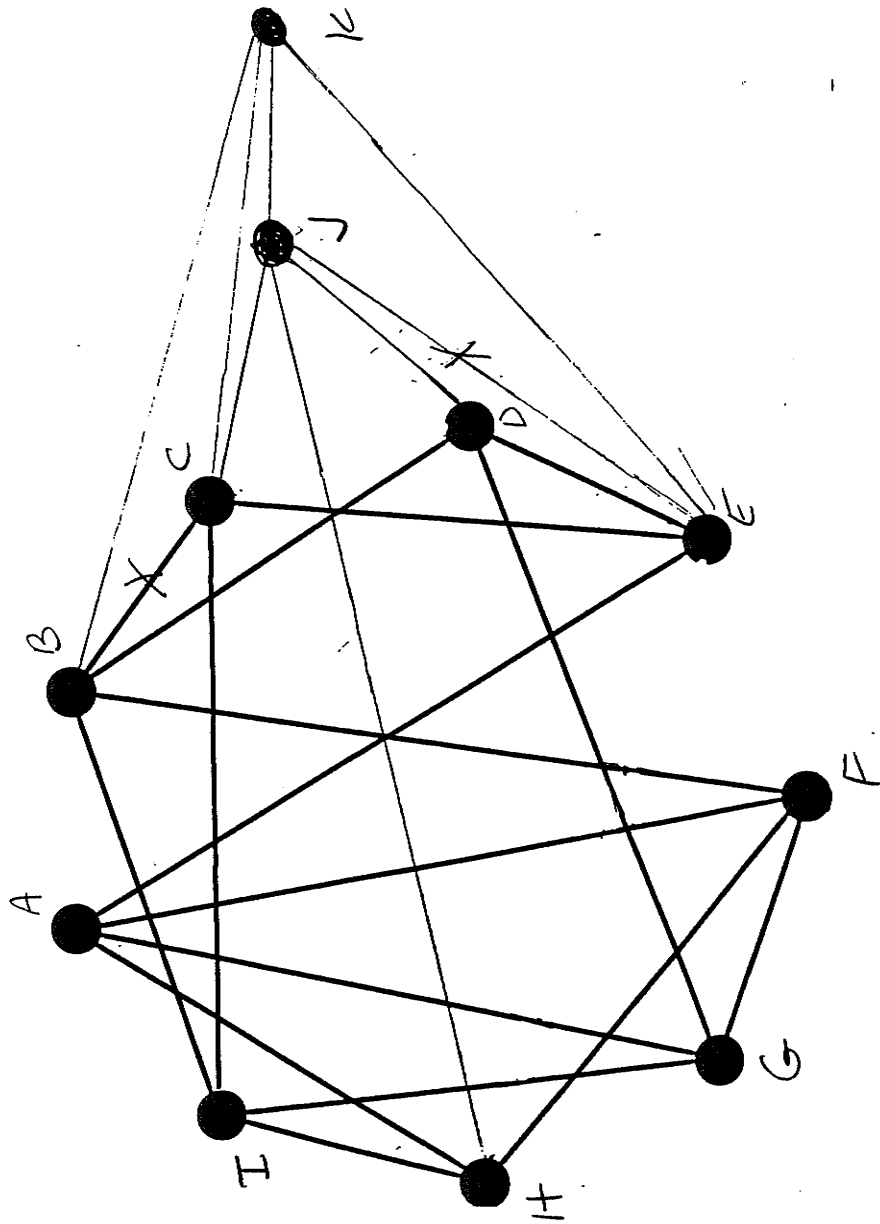


FIG 4B

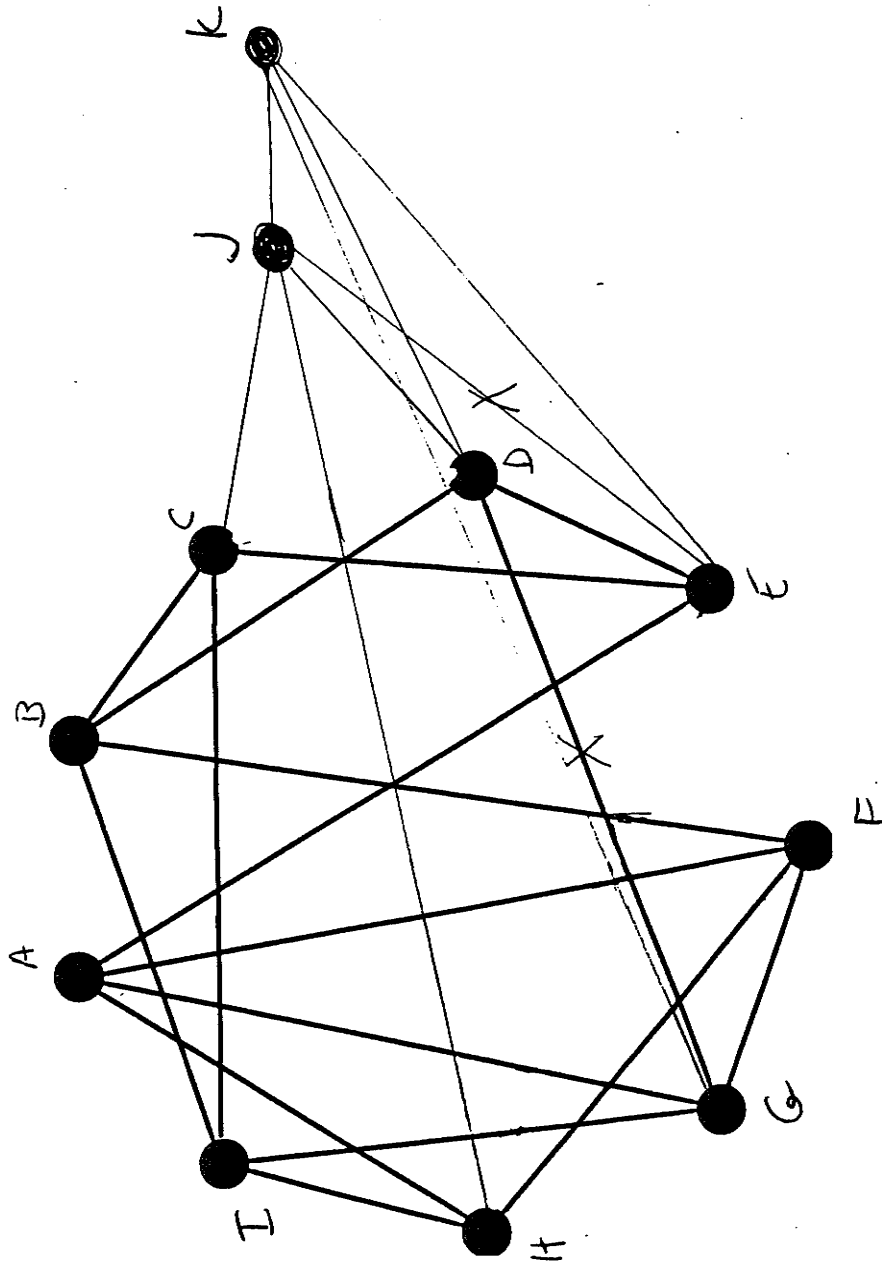


Fig 4C

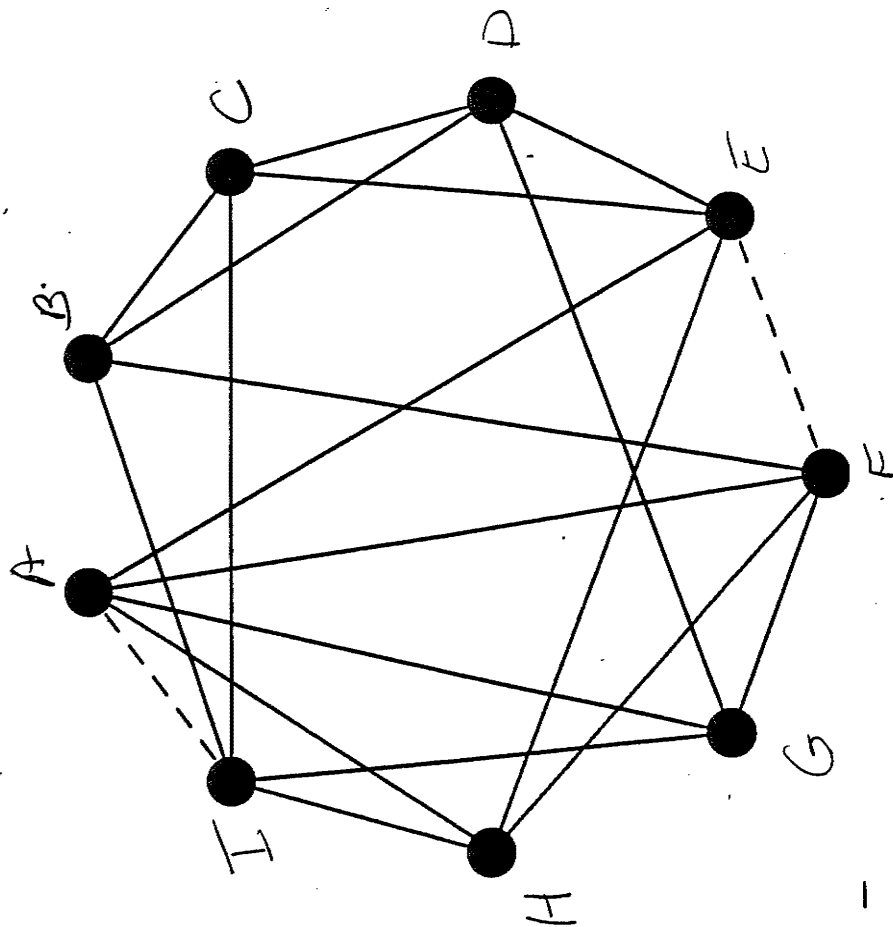


Fig 5A

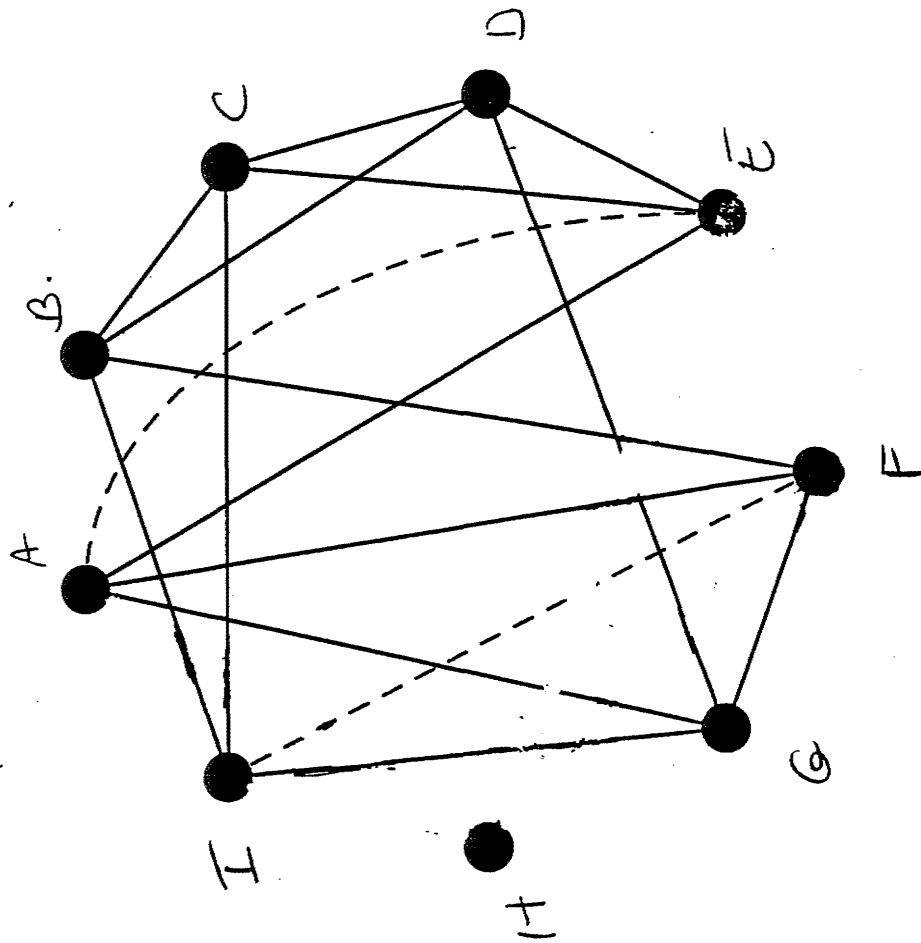


Fig 5B

2025 RELEASE UNDER E.O. 14176

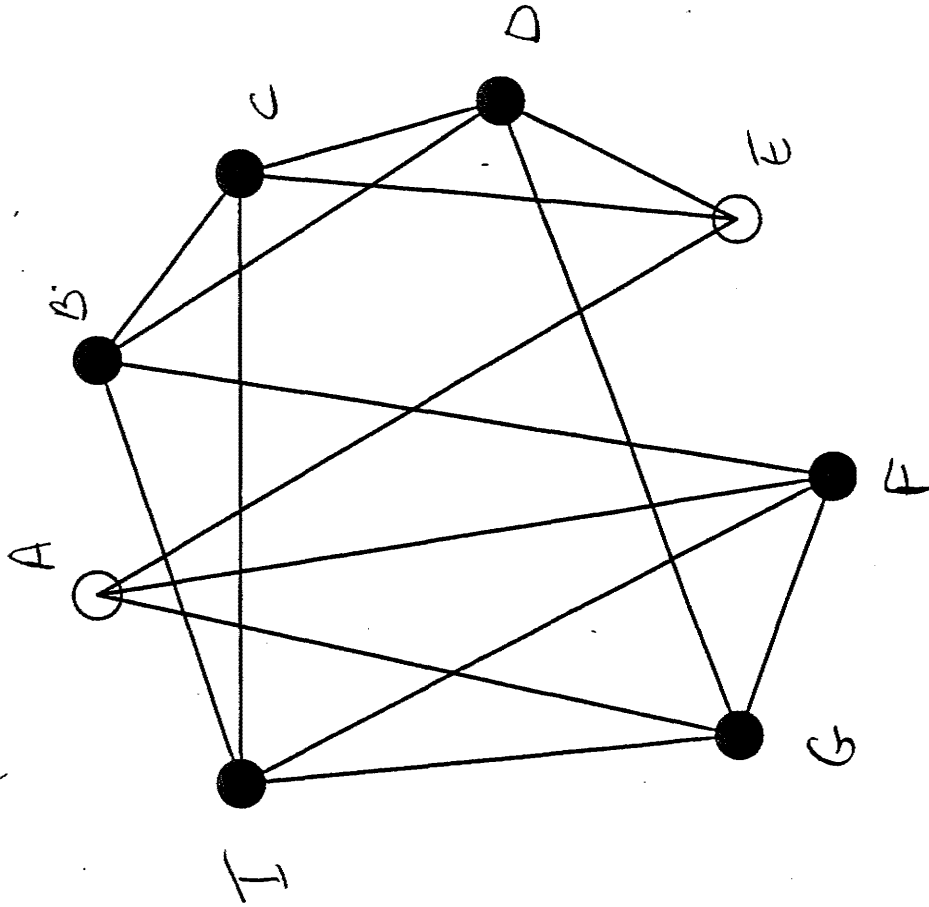


FIG 5C

UNCLASSIFIED

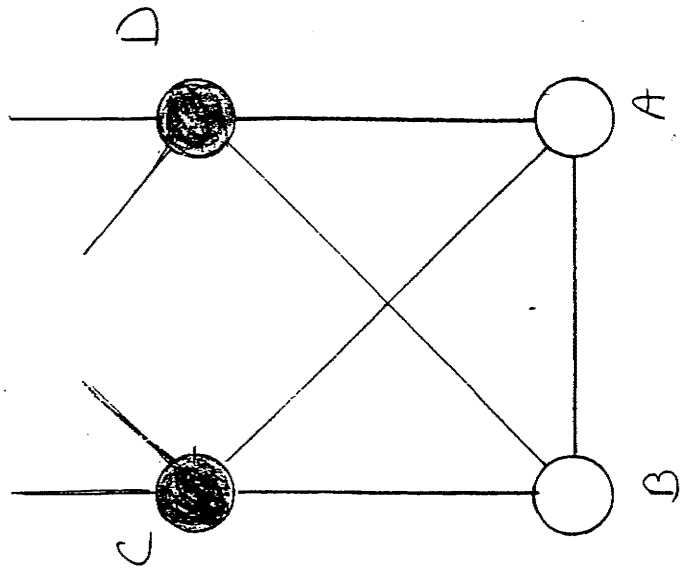


Fig 5F

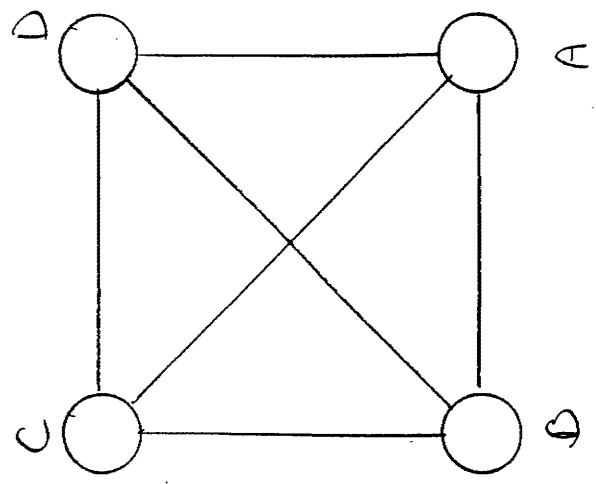


Fig 5E

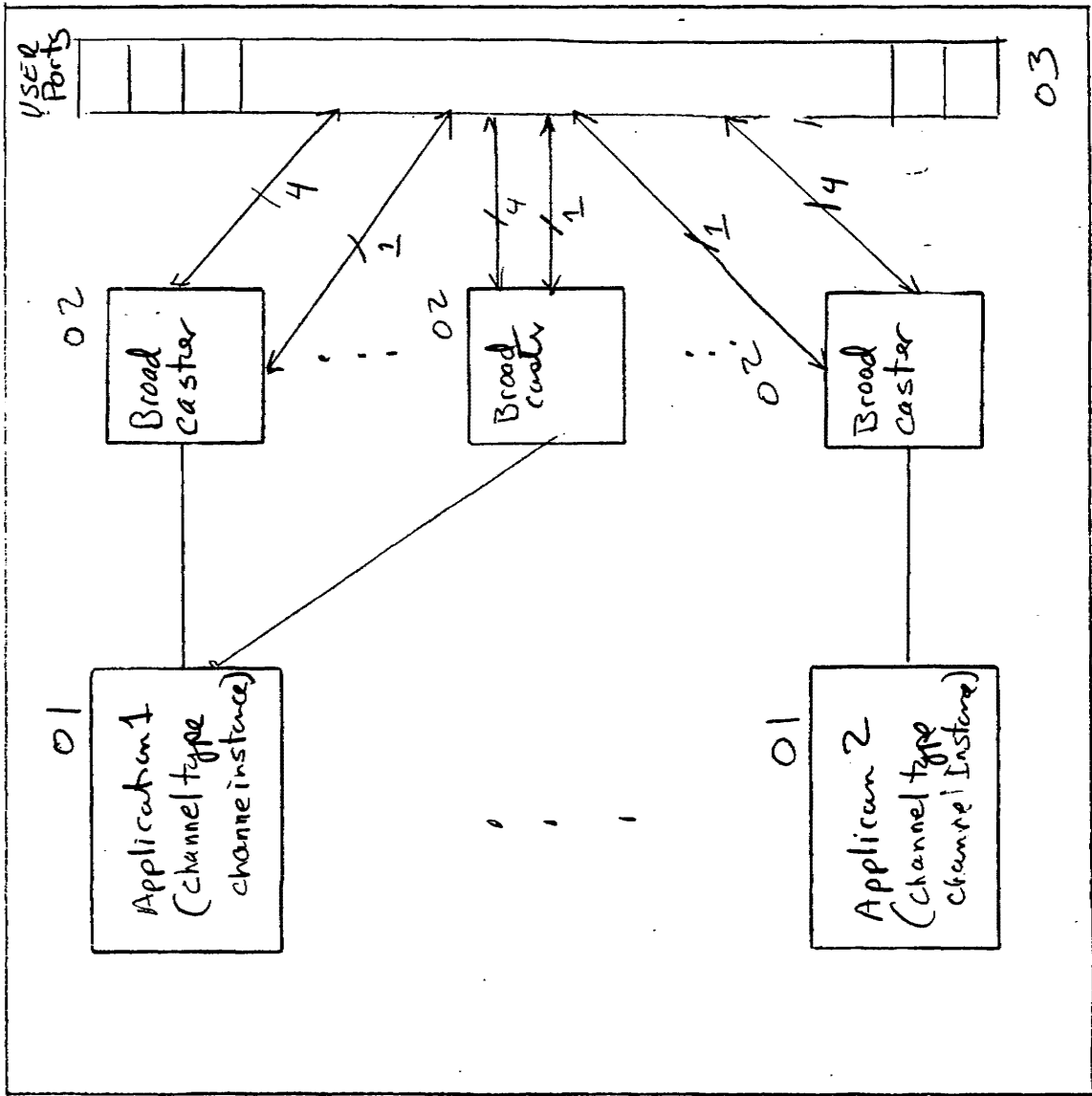


Fig 4

(Channel Type,
Channel Instance,
ConnectAuxInfo)

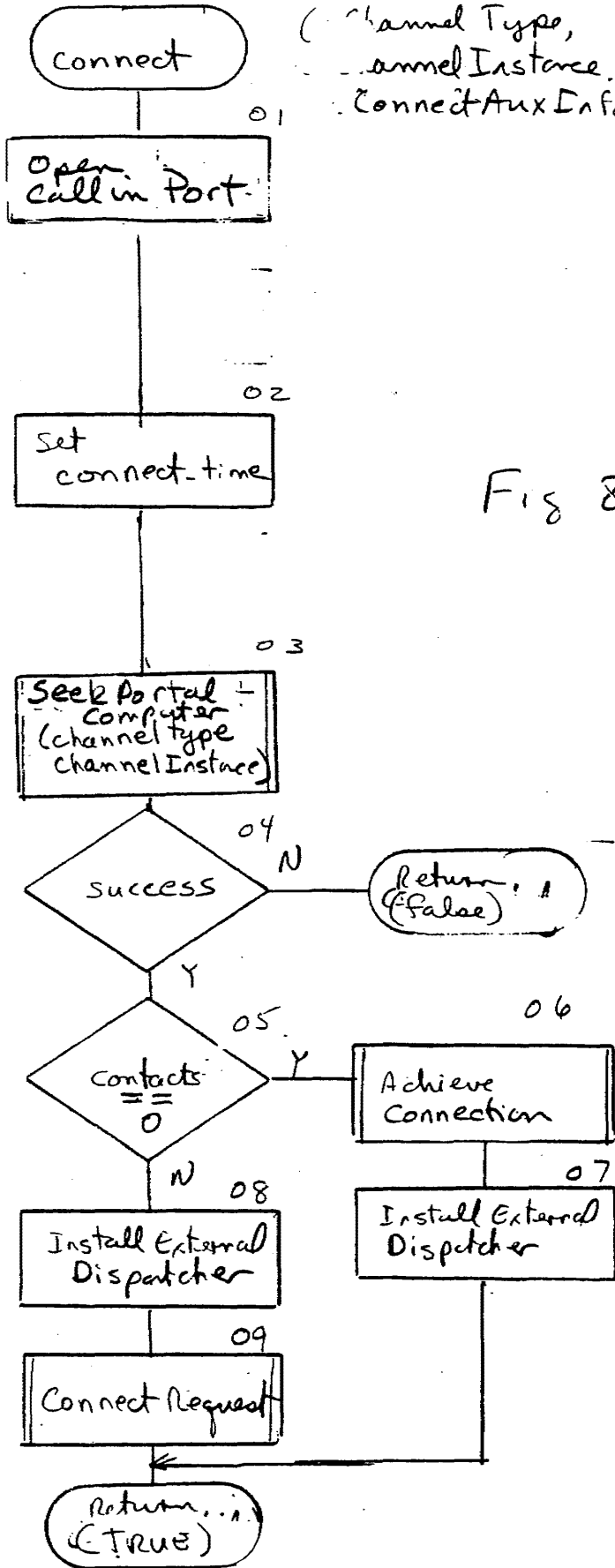


Fig 8

00000000000000000000

Request

channel type
channel Instance

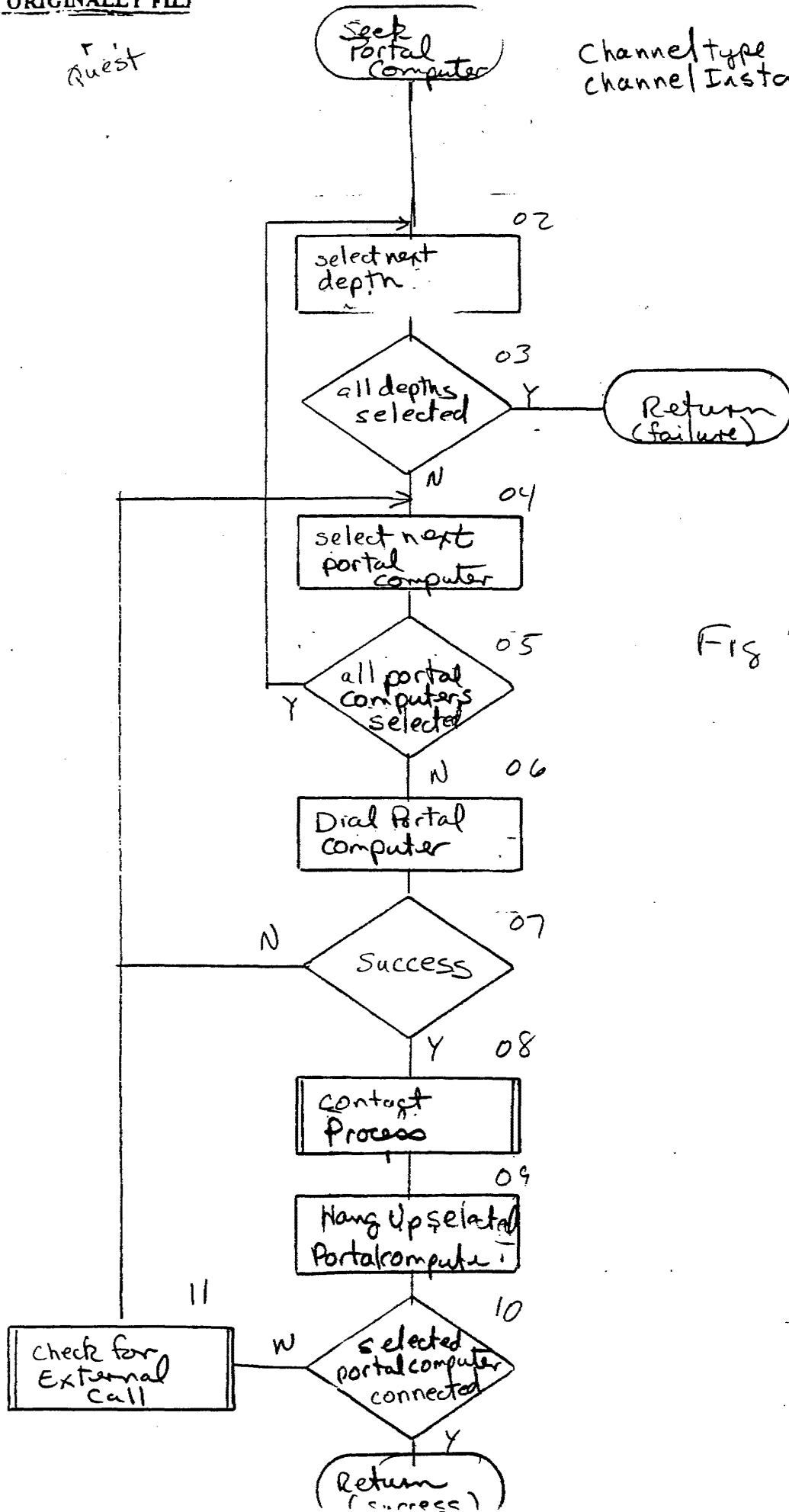
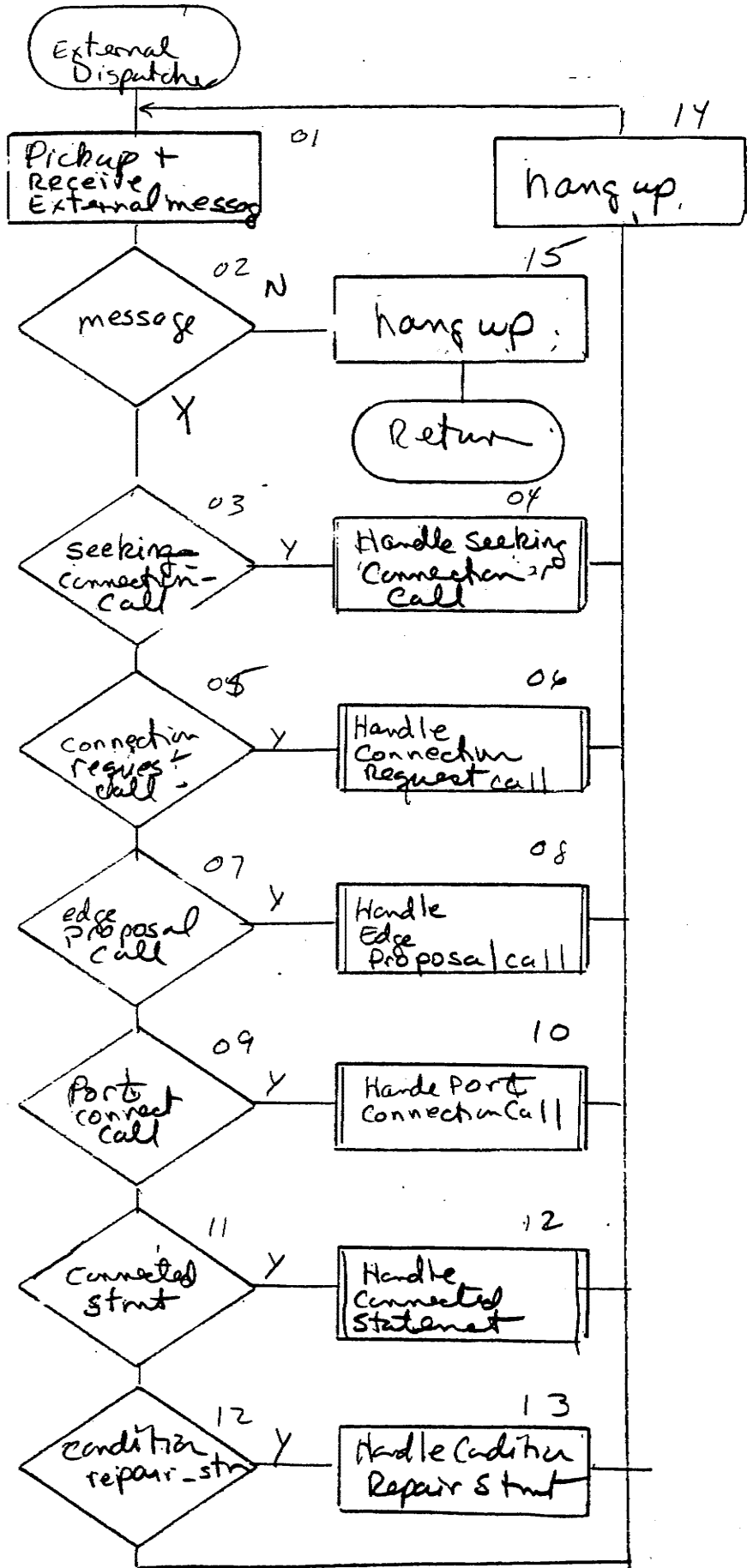


Fig 9

p 1

Fig 14



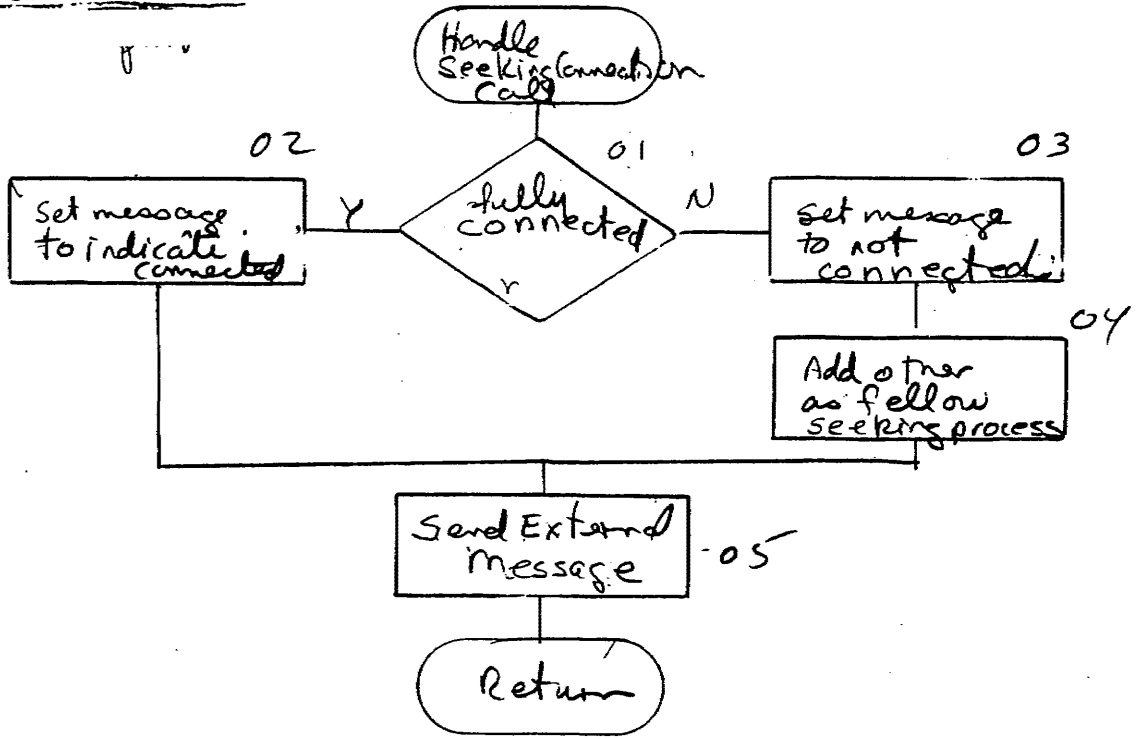
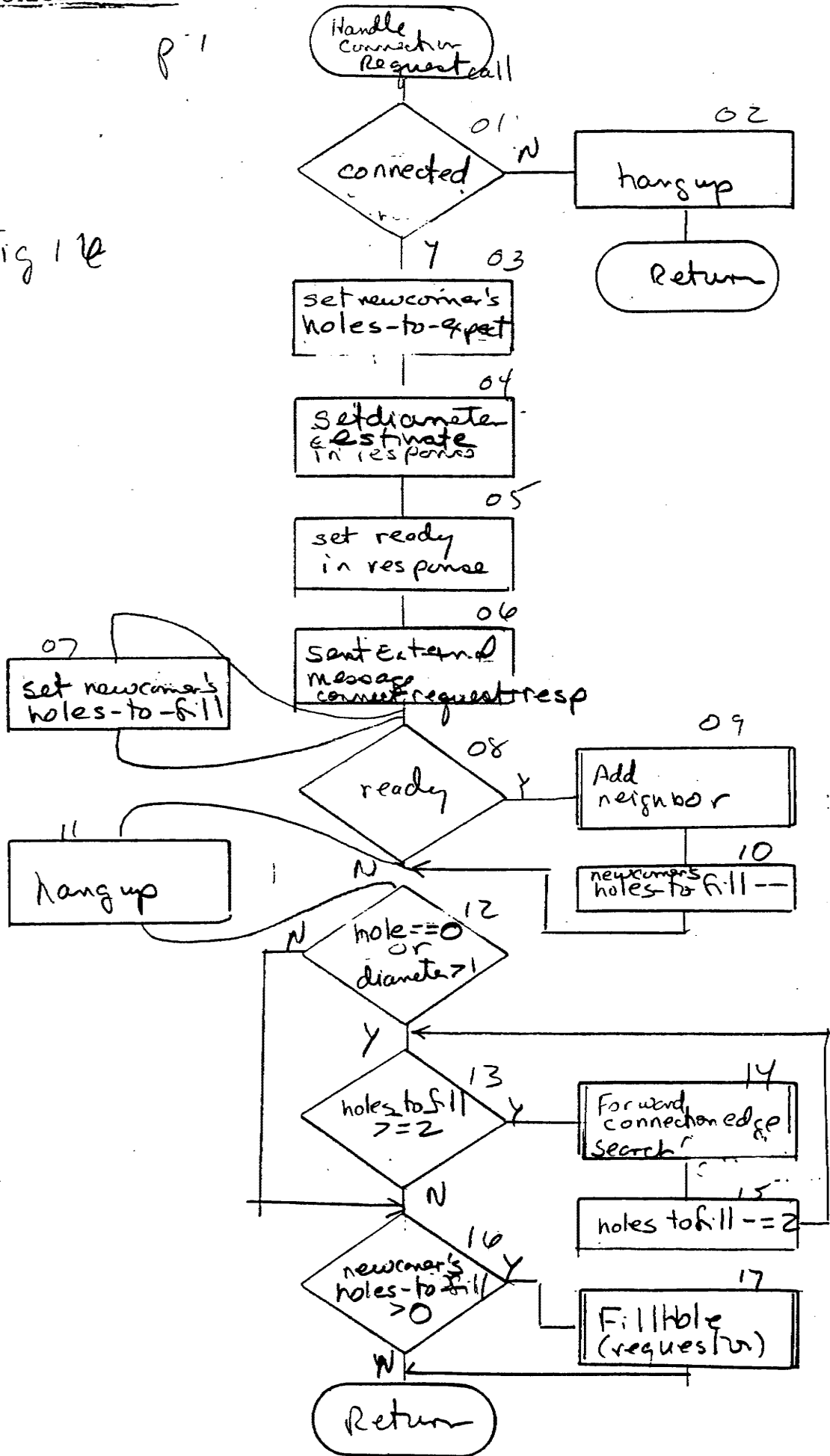


Fig 15

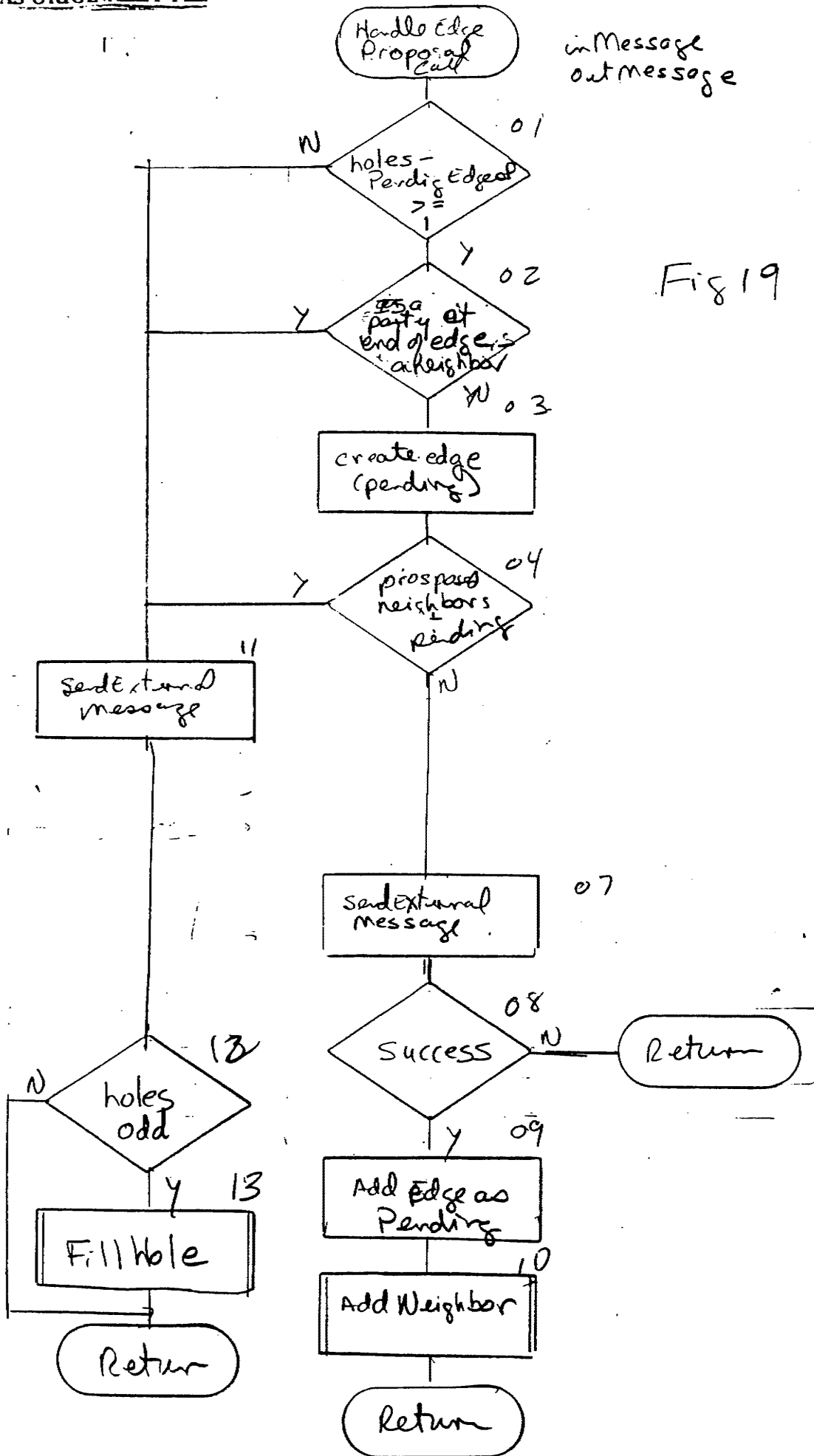
P-1

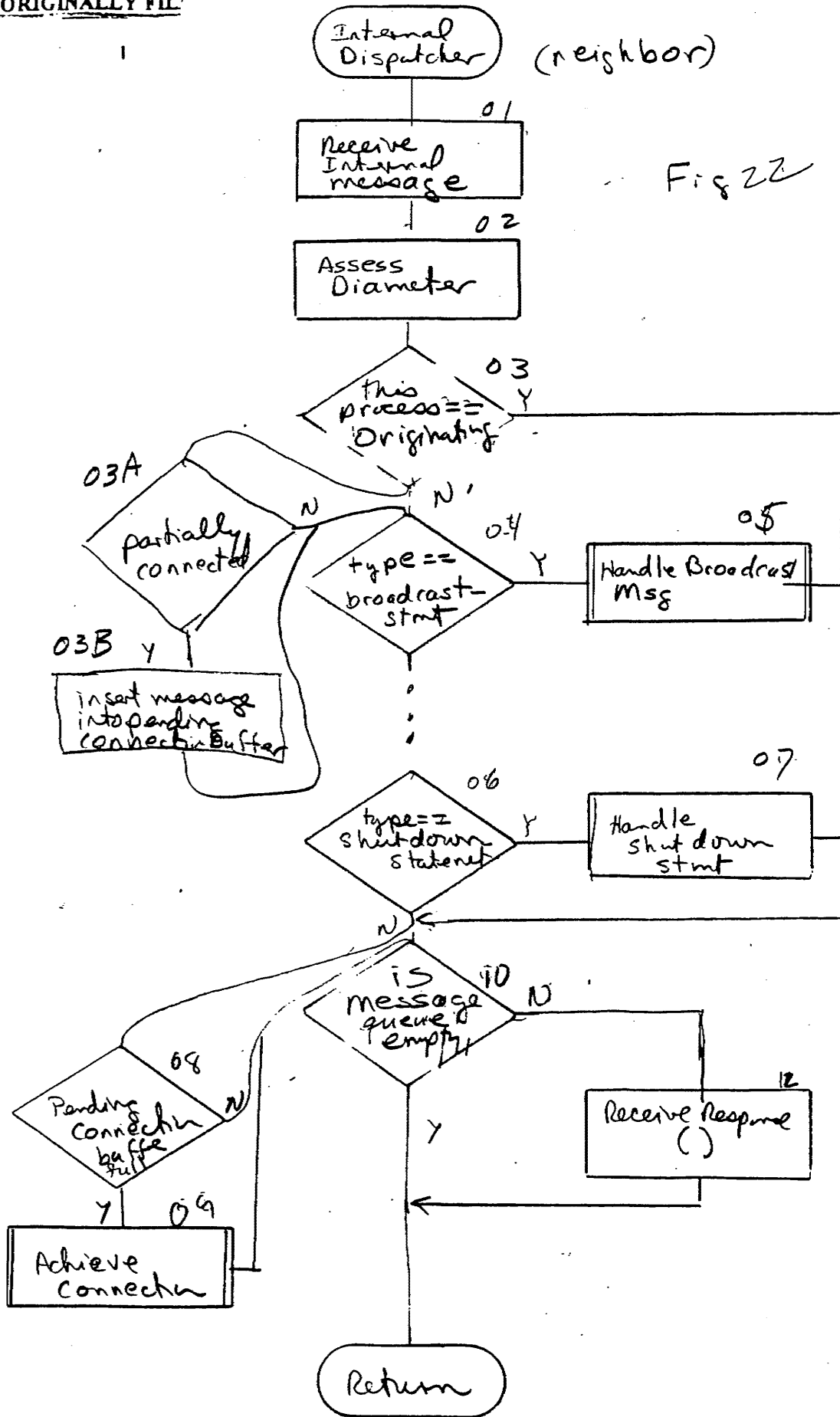
Fig 11a

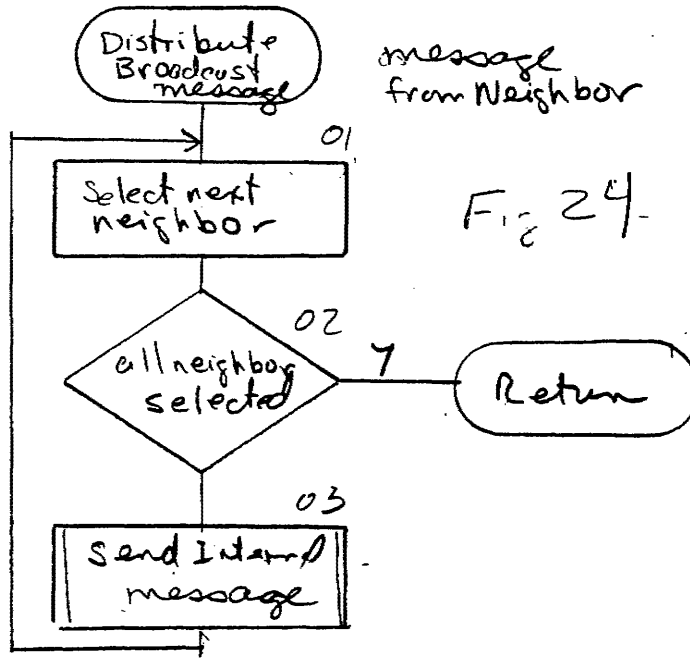


inMessage
outMessage

Fig 19







01
02
03

from Neighbor
message

Fig 28

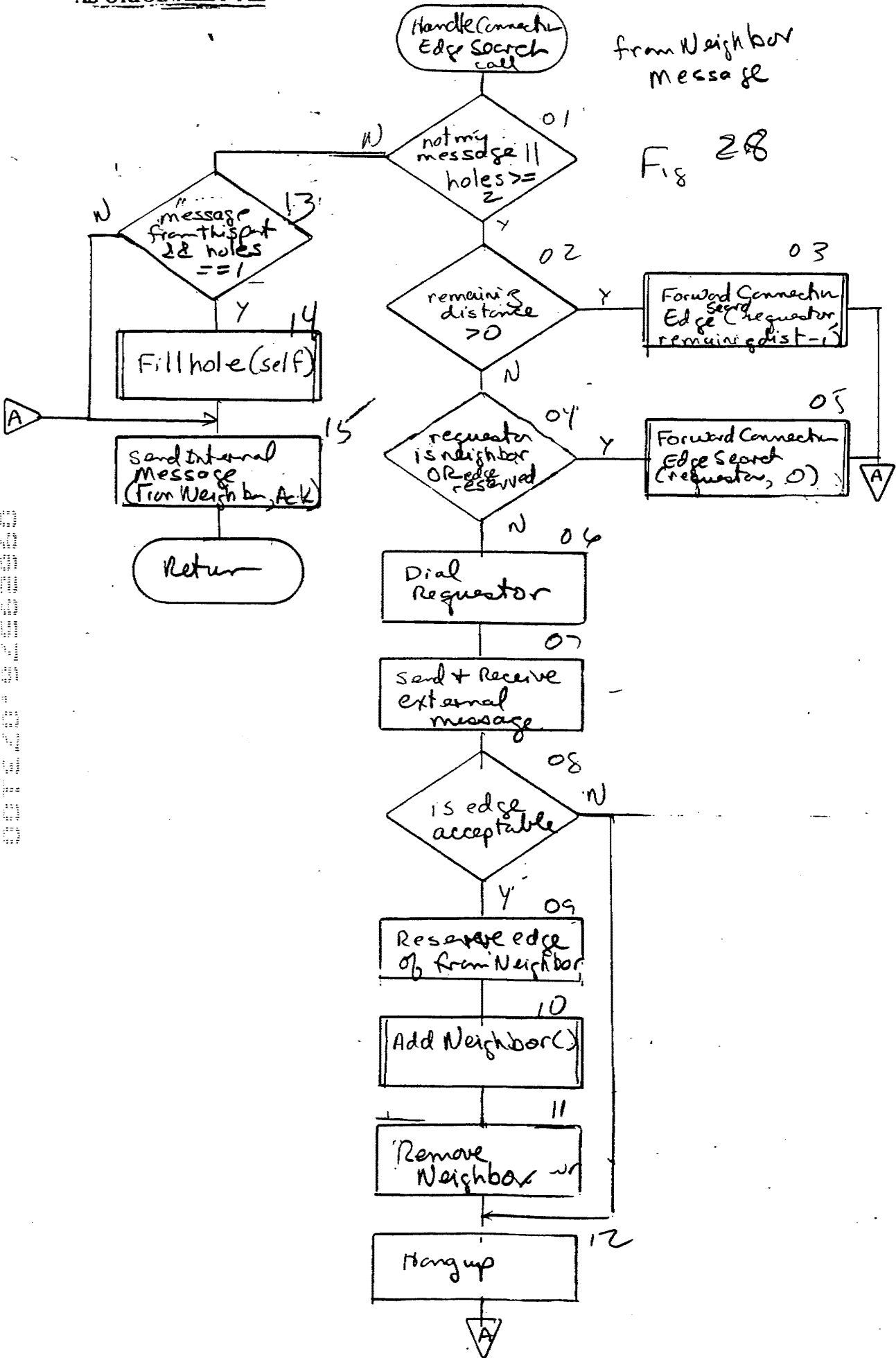
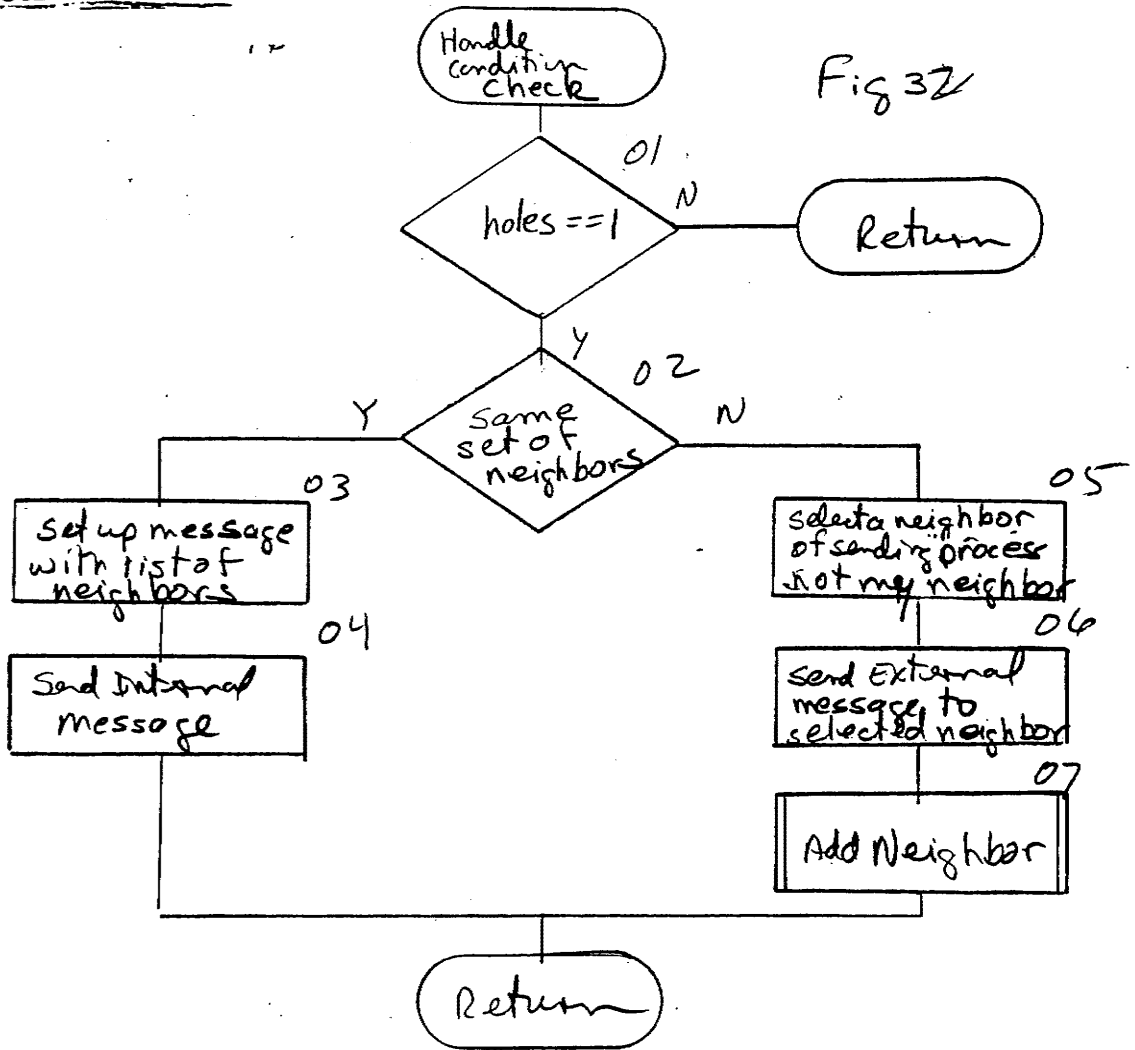
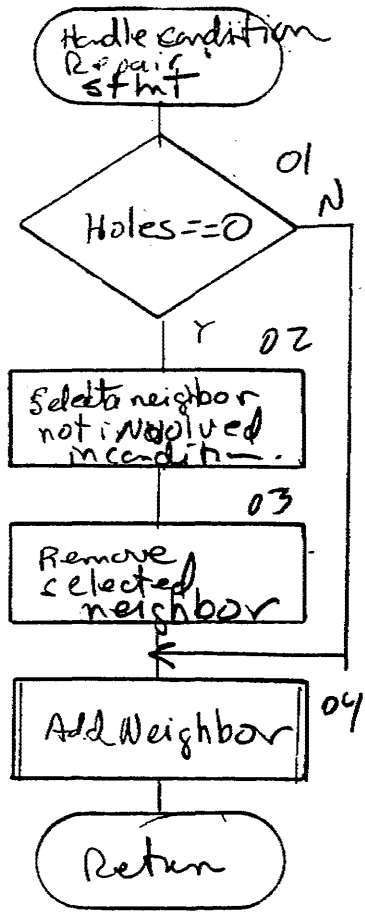


Fig 32



23

Fig 33



001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100

2



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
 UNITED STATES PATENT AND TRADEMARK OFFICE
 WASHINGTON, D.C. 20231
 www.uspto.gov

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/629,576	07/31/2000	Virgil E. Bourassa	030048001

25096
 PERKINS COIE LLP
 PATENT-SEA
 PO BOX 1247
 SEATTLE, WA 98111-1247

FORMALITIES LETTER



OC00000005422659

Date Mailed: 09/25/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
Applicant must submit \$ 690 to complete the basic filing fee and/or file a small entity statement claiming such status (37 CFR 1.27).
- Total additional claim fee(s) for this application is \$756.
 - \$522 for 29 total claims over 20.
 - \$234 for 3 independent claims over 3 .
- The oath or declaration is missing.
A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of \$130 for a non-small entity, must be submitted with the missing items identified in this letter.

- The balance due by applicant is \$ 1576.

*A copy of this notice **MUST** be returned with the reply.*

Customer Service Center
 Initial Patent Examination Division (703) 308-1202
 PART 3 - OFFICE COPY

sector

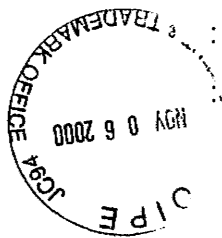
PATENT

I hereby certify that on the date specified below, this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to Box Missing Parts, Commissioner for Patents, Washington, DC 20231.

10/30/10
Date
Jeanne Connelly
Jeanne Connelly

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants : Fred B. Holt and Virgil E. Bourassa
Application No. : 09/629,576
Filed : July 31, 2000
For : BROADCASTING NETWORK



Docket No. : 030048001US
Date : October 30, 2000

Box Missing Parts
Commissioner for Patents
Washington, DC 20231

RESPONSE TO NOTICE TO FILE MISSING PARTS OF APPLICATION

Sir:

In response to the Notice to File Missing Parts dated September 25, 2000, please find enclosed a Declaration, Power of Attorney, Authorization for Extensions of Time Under 37 CFR § 1.136(a)(3), and copy of the Notice to File Missing Parts for the above-identified application.

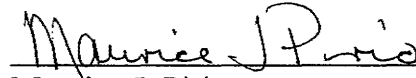
The fees have been calculated as follows:

Basic Fee	\$	710.00
Total Claims (49, 29 extra)		522.00
Independent Claims (6, 3 extra)		240.00
Missing Parts Surcharge		130.00
Total	\$	1602.00

The Commissioner is hereby authorized to charge the filing fees, and any additional filing fees or to credit any overpayment to Deposit Account No. 50-0665. A duplicate copy of this response is enclosed.

Respectfully submitted,

Perkins Coie LLP



Maurice J. Pirio

Registration No. 33,273

MJP:jc

Enclosures:

Postcard

Copy of this Response

Declaration

Power of Attorney

Authorization for Extensions of Time Under 37 CFR § 1.136(a)(3)

Copy of Notice to File Missing Parts

PERKINS COIE LLP

P.O. Box 1247

Seattle, Washington 98111-1247

(206) 583-8888

FAX: (206) 583-8500



7/2

DECLARATION

As the below-named inventors, we declare that:

Our residences, post office addresses, and citizenships are as stated below under our names.

We believe we are the original, first, and joint inventors of the subject matter claimed and for which a patent is sought on the invention entitled "BROADCASTING NETWORK," the specification of which was filed in the U.S. Patent and Trademark Office on July 31, 2000 and assigned application number 09/629,576.

We have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to above.

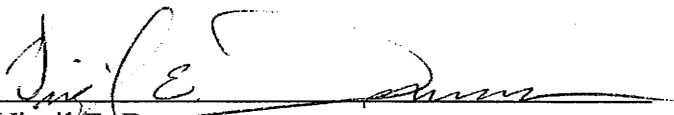
We acknowledge our duty to disclose information which is material to the patentability of this application in accordance with 37 C.F.R. § 1.56(a).

We further declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that the making of willfully false statements and the like is punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and may jeopardize the validity of any patent issuing from this patent application.

Fred B. Holt

Date 26 Oct 2000

Residence : City of Seattle
State of Washington
Citizenship : United States of America
P.O. Address : 5520 31st Avenue NE
Seattle, Washington 98105


Virgil E. Bourassa

Date 10/26/2000

Residence : City of Bellevue
State of Washington
Citizenship : United States of America
P.O. Address : 16110 SE 24th Street
Bellevue, Washington 98008

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants : Fred B. Holt and Virgil E. Bourassa
 Application No. : 09/629,576
 Filed : July 31, 2000
 For : BROADCASTING NETWORK



Art Unit : 2731
 Docket No. : 030048001US

Commissioner for Patents
 Washington, DC 20231

ELECTION UNDER 37 C.F.R. §§ 3.71 AND 3.73
AND POWER OF ATTORNEY

Sir:

The undersigned, being Assignee of the entire interest in the above-identified application by virtue of an Assignment filed concurrently herewith, a copy of which is enclosed, hereby elects under 37 C.F.R. § 3.71, to prosecute the application to the exclusion of the inventors.

Assignee hereby appoints JERRY A. RIEDINGER, Registration No. 30,582; MAURICE J. PIRIO, Registration No. 33,273; JOHN C. STEWART, Registration No. 40,188; MICHAEL D. BROADDUS, Registration No. 41,637; BRIAN P. MCQUILLEN, Registration No. 41,989; CATHERINE HONG TRAN, Registration No. 43,960; ROBERT G. WOOLSTON, Registration No. 37,263; PAUL T. PARKER, Registration No. 38,264; JOHN M. WECHKIN, Registration No. 42,216; CHRISTOPHER DALEY-WATSON, Registration No. 34,807; STEVEN D. LAWRENZ, Registration No. 37,376; JAMES A.D. WHITE, Registration No. 43,985; and FRANK ABRAMONTE, Registration No. 38,066, of the firm of Perkins Coie LLP and ROBERT

L. GULLETTE, Registration No. 26,899, PAUL C. CULLOM, JR., Registration No. 25,580, ANN K. GALBRAITH, Registration No. 33,530, JAMES P. HAMLEY, Registration No. 28,081, JOHN C. HAMMAR, Registration No. 29,928, LAWRENCE W. NELSON, Registration No. 34,684 and ROBERT R. RICHARDSON, Registration No. 40,143 of The Boeing Company, as the principal attorneys with full power of substitution, association, and revocation to prosecute said application, to transact all business in the Patent and Trademark Office connected therewith, and to receive the letters patent therefor. Please direct all telephone calls to Maurice J. Pirio at (206) 583-8888 and telecopies to (206) 583-8500.

Please direct all correspondence to:

Patent-SEA
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
Attn: Maurice J. Pirio

Pursuant to 37 C.F.R. § 3.73, the undersigned duly authorized designee of Assignee certifies that the evidentiary documents have been reviewed, specifically the Assignment to The Boeing Company filed concurrently herewith for recording, a copy of which is attached hereto, and certifies that to the best of my knowledge and belief, title remains in the name of the Assignee.

The Boeing Company

10/27/00
Date

Robert R. Richardson
Name of Person Signing

Counsel
Title of Person Signing

MJP:jc

Enclosure:
Copy of Assignment

ASSIGNMENT

WHEREAS, we, Fred B. Holt and Virgil E. Bourassa ("ASSIGNORS"), having post office addresses of 5520 31st Avenue NE, Seattle, Washington 98105 and 16110 SE 24th Street, Bellevue, Washington 98008, respectively, are the joint inventors of an invention entitled "BROADCASTING NETWORK," as described and claimed in the specification for which an application for United States letters patent was filed on July 31, 2000 and assigned Application No. 09/629,576.

WHEREAS, The Boeing Company ("ASSIGNEE"), a corporation of the State of Delaware having its principal place of business at Seattle, Washington, is desirous of acquiring the entire right, title, and interest in and to the invention and in and to any patents that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, ASSIGNORS hereby sell, assign, and transfer unto ASSIGNEE, its legal representatives, successors, and assigns, the entire right, title and interest in and to the invention as set forth in the above-mentioned application, including any continuations, continuations-in-part, divisions, reissues, re-examinations, or extensions thereof, any other inventions described in the application, and any and all patents of the United States of America and all foreign countries that may be issued for the invention, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from the United States application to which foreign applications are entitled by virtue of international convention, treaty or otherwise, the invention, application and all patents on the invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNORS had this assignment, transfer, and sale not been made.

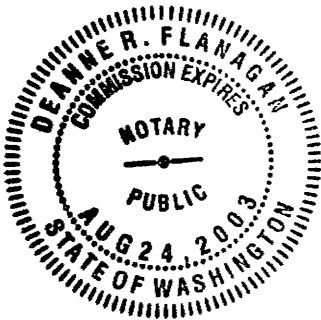
UPON THE ABOVE-STATED CONSIDERATIONS, ASSIGNORS agree to not execute any writing or do any act whatsoever conflicting with this assignment, and at any time upon request, without further or additional consideration but at the expense of ASSIGNEE, execute all instruments and documents and do such additional acts as ASSIGNEE may deem necessary or desirable to perfect ASSIGNEE's enjoyment of this grant, and render all necessary assistance required for the making and prosecution of applications for United States and foreign patents on the invention, for litigation regarding the patents, or for the purpose of protecting title to the invention or patents therefor.

ASSIGNORS authorize and request the Commissioner of Patents and Trademarks to issue any Patent of the United States that may be issued for the invention to ASSIGNEE.

26 Oct 2000
Date Fred B. Holt

State of Washington)
County of King) ss.

I certify that I know or have satisfactory evidence that Fred B. Holt is the person who appeared before me, and the person acknowledged that he signed this instrument and acknowledged it to be his free and voluntary act for the uses and purposes mentioned in the instrument.



Dated 10.26.00
Signature of Deanne R. Flanagan
Notary Public
Printed Name Deanne R. Flanagan
My appointment expires 8-24-03

10/26/2000
Date

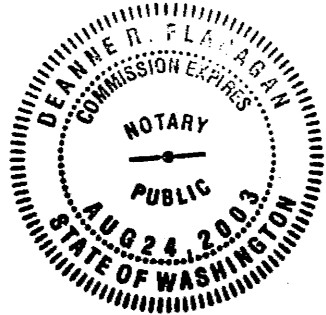
Virgil E. Bourassa
Virgil E. Bourassa

State of Washington)

County of King)

ss.

I certify that I know or have satisfactory evidence that Virgil E. Bourassa is the person who appeared before me, and the person acknowledged that he signed this instrument and acknowledged it to be his free and voluntary act for the uses and purposes mentioned in the instrument.



Dated 10.26.2000

Signature of Notary Public Deanne R. Flanagan

Printed Name Deanne R. Flanagan

My appointment expires 8.24.03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants : Fred B. Holt and Virgil E. Bourassa

Application No. : 09/629,576

Filed : July 31, 2000

For : BROADCASTING NETWORK



Art Unit : 2731

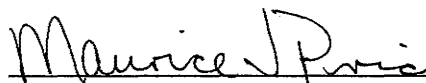
Docket No. : 030048001US

Date : October 30, 2000

Commissioner for Patents
Washington, DC 20231AUTHORIZATION FOR EXTENSIONS OF TIME UNDER 37 C.F.R. § 1.136(A)(3)

Sir:

With respect to the above-identified application, the Commissioner is authorized to treat any concurrent or future reply requiring a petition for an extension of time under 37 C.F.R. § 1.136(a)(3) for its timely submission as incorporating a petition therefor for the appropriate length of time. The Commissioner is also authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account No. 50-0665.

Date 10/30/00

Maurice J. Pirio

Registration No. 33,273

PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 583-8888
FAX: (206) 583-8500



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/629,576	07/31/2000	Virgil E. Bourassa	030048001

25096
PERKINS COIE LLP
PATENT-SEA
PO BOX 1247
SEATTLE, WA 98111-1247

FORMALITIES LETTER



Date Mailed: 09/25/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
Applicant must submit \$ 690 to complete the basic filing fee and/or file a small entity statement claiming such status (37 CFR 1.27).
- Total additional claim fee(s) for this application is \$756.
 - \$522 for 29 total claims over 20.
 - \$234 for 3 independent claims over 3 .
- The oath or declaration is missing.
A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of \$130 for a non-small entity, must be submitted with the missing items identified in this letter.
- The balance due by applicant is \$ 1576.

A copy of this notice MUST be returned with the reply.

Customer Service Center
Initial Patent Examination Division (703) 308-1202

PART 2 - COPY TO BE RETURNED WITH RESPONSE

09629576

11/05/2000 KZEWIE 00000001 500655

710.00 CH
522.00 CH
240.00 CH
130.00 CH

01 FD:101
02 FD:103
03 FD:102
04 FD:105



UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

ASSISTANT SECRETARY AND COMMISSIONER
OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

RECEIVED

MAY 3 - 2001

Technology Center 2100

CHANGE OF ADDRESS/POWER OF ATTORNEY

FILE LOCATION 2152 SERIAL NUMBER 09629576 PATENT NUMBER

THE CORRESPONDENCE ADDRESS HAS BEEN CHANGED TO CUSTOMER # 25096

THE PRACTITIONERS OF RECORD HAVE BEEN CHANGED TO CUSTOMER # 25096

THE FEE ADDRESS HAS BEEN CHANGED TO CUSTOMER # 25096

ON 04/12/01 THE ADDRESS OF RECORD FOR CUSTOMER NUMBER 25096 IS:

PERKINS COIE LLP
1201 3RD AVENUE , SUITE 4800
SEATTLE WA 98101-3099

AND THE PRACTITIONERS OF RECORD FOR CUSTOMER NUMBER 25096 ARE:

30582	33273	33904	34807	37263	37376	38264	40188	41637	41989
42216	43960	43985	46140						

PTO INSTRUCTIONS: PLEASE TAKE THE FOLLOWING ACTION WHEN THE CORRESPONDENCE ADDRESS HAS BEEN CHANGED TO CUSTOMER NUMBER: RECORD, ON THE NEXT AVAILABLE CONTENTS LINE OF THE FILE JACKET, 'ADDRESS CHANGE TO CUSTOMER NUMBER'. LINE THROUGH THE OLD ADDRESS ON THE FILE JACKET LABEL AND ENTER ONLY THE 'CUSTOMER NUMBER' AS THE NEW ADDRESS. FILE THIS LETTER IN THE FILE JACKET. WHEN ABOVE CHANGES ARE ONLY TO FEE ADDRESS AND/OR PRACTITIONERS OF RECORD, FILE LETTER IN THE FILE JACKET. THIS FILE IS ASSIGNED TO GAU 2731.

09/2755

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, Washington, D.C., 20231, on:

Date: 4/18/02

By: *Jeanne Connelly*
Jeanne Connelly #5



PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:

VIRGIL E. BOURASSA AND FRÉD B. HOLT

APPLICATION NO.: 09/629,576

FILED: JULY 31, 2000

FOR: BROADCASTING NETWORK

RECEIVED

APR 24 2002

Technology Center 2100

Information Disclosure Statement Within Three Months of Application Filing or Before First Action – 37 CFR 1.97(b)

Commissioner for Patents
Washington, D.C. 20231

Sir:

1. Timing of Submission

This information disclosure is being filed within three months of the filing date of this application or date of entry into the national stage of an international application or before the mailing date of a first Office action on the merits, whichever occurs last [37 CFR 1.97(b)]. The references listed on the enclosed Form PTO/SB/08A (modified) may be material to the examination of this application; the Examiner is requested to make them of record in the application.

2. Cited Information

Copies of the following references are enclosed:

- All cited references
- References marked by asterisks
- The following:

Copies of the following references can be found in parent application Ser. No.

- All cited references
- References marked by asterisks
- The following:

The following references are not in English. For each such reference, the undersigned has enclosed (i) a translation of the reference; (ii) a copy of a communication from a foreign patent office or International Searching Authority citing the reference, (iii) a copy of a reference which appears to be an English-language counterpart, or (iv) an English-language abstract for the reference prepared by a third party. Applicant has not verified that the



translation, English-language counterpart or third-party abstract is an accurate representation of the teachings of the non-English reference, though, and reserves the right to demonstrate otherwise.

- All cited references
- References marked by ampersands
- The following:

3. Effect of Information Disclosure Statement (37 CFR 1.97(h))

This Information Disclosure Statement is not to be construed as a representation that: (i) a search has been made; (ii) additional information material to the examination of this application does not exist; (iii) the information, protocols, results and the like reported by third parties are accurate or enabling; or (iv) the cited information is, or is considered to be, material to patentability. In addition, applicant does not admit that any enclosed item of information constitutes prior art to the subject invention and specifically reserves the right to demonstrate that any such reference is not prior art.

4. Fee Payment

No fees are believed due. However, should the Commissioner determine that fees are due in order for this Information Disclosure Statement to be considered, the Commissioner is hereby authorized to charge such fees to Deposit Account No. 50-0665.

5. Patent Term Adjustment (37 CFR 1.704(d))

- The undersigned states that each item of information submitted herewith was cited in a communication from a foreign patent office in a counterpart application and that this communication was not received by any individual designated in 37 C.F.R. § 1.56(c) more than thirty days prior to the filing of this statement. 37 C.F.R. § 1.704(d).

Respectfully submitted,
Perkins Coie LLP

Maurice J. Pirio

Registration No. 33,273

Date: 4-18-02

Correspondence Address:

Customer No. 25096
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
Phone: (206) 583-8888

RECEIVED

APR 24 2002

Technology Center 2100

Please type a plus sign (+) inside this box →

<p>Substitute for form 1449A/PTO</p> <p>INFORMATION DISCLOSURE STATEMENT BY APPLICANT (use as many sheets as necessary)</p> <p>APR 23 2002</p> <p>TRADEMARK OFFICE</p>		COMPLETE IF KNOWN	
		Application Number	09/629,576
		Confirmation Number	
		Filing Date	July 31, 2000
		First Named Inventor	Virgil E. Bourassa #5
		Group Art Unit	2155
Examiner Name	WON		
Sheet 1 of 5	Attorney Docket No.	030048001US	

U.S. PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		NUMBER	Kind Code (if known)			
yu	AB	09/629,570		Bourassa et al.	7/31/00	
yu	AC	09/629,577		Bourassa et al.	7/31/00	
yu	AD	09/629,575		Bourassa et al.	7/31/00	
yu	AE	09/629,572		Bourassa et al.	7/31/00	
yu	AF	09/629,023		Bourassa et al.	7/31/00	
yu	AG	09/629,043		Bourassa et al.	7/31/00	
yu	AH	09/629,024		Bourassa et al.	7/31/00	
yu	AI	09/629,042		Bourassa et al.	7/31/00	
yu	AJ	6,304,928		Mairs et al.	10/16/01	
yu	AK	6,285,363		Mairs et al.	9/4/01	
yu	AL	6,271,839		Mairs et al.	8/7/01	
yu	AM	6,268,855		Mairs et al.	7/31/01	
yu	AN	6,243,691		Fisher et al.	6/5/01	
yu	AO	6,223,212		Batty et al.	4/24/01	
yu	AP	6,216,177		Mairs et al.	4/10/01	
yu	AQ	6,199,116		May et al.	3/6/01	
yu	AR	6,094,676		Gray et al.	7/25/00	

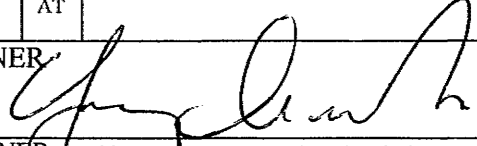
RECEIVED
APR 24 2002
Technology Center 2100

FOREIGN PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	Foreign Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		Office	Number				
	AS						

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

EXAMINER INITIALS	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume/issue number(s), publisher, city and/or country where published.	T
	AT		

EXAMINER: 

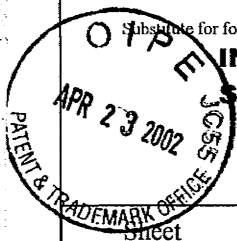
DATE CONSIDERED
1-29-04

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

Please type a plus sign (+) inside this box → +

PTO/SB/08A

Approved for use through 10/31/99. OMB 0651-0031
 Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE



Substitute for form 1449A/PTO

**INFORMATION DISCLOSURE
 STATEMENT BY APPLICANT**

(use as many sheets as necessary)

COMPLETE IF KNOWN

Application Number	09/629,576
Confirmation Number	
Filing Date	July 31, 2000
First Named Inventor	Virgil E. Bourassa
Group Art Unit	2155 #5
Examiner Name	WON
Attorney Docket No.	030048001US

Sheet 2 of 5

U.S. PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		NUMBER	Kind Code (if known)			
yu	AA	6,047,289		Thorne et al.	4/4/00	
yu	AB	6,038,602		Ishikawa	3/14/00	
yu	AC	6,032,188		Mairs et al.	2/29/00	
yu	AD	6,029,171		Smiga et al.	2/22/00	
yu	AE	6,023,734		Ratcliff et al.	2/8/00	
yu	AF	6,013,107		Blackshear et al.	1/11/00	
yu	AG	6,003,088		Houston et al.	12/14/99	
yu	AH	5,987,506		Carter et al.	11/16/99	
yu	AI	5,974,043		Solomon	10/26/99	
yu	AJ	5,956,484		Rosenberg et al.	9/21/99	
yu	AK	5,948,054		Nielsen	9/7/99	
yu	AL	5,949,975		Batty et al.	9/7/99	
yu	AM	5,935,215		Bell et al.	8/10/99	
yu	AN	5,928,335		Morita	7/27/99	
yu	AO	5,907,610		Onweller	5/25/99	
yu	AP	5,899,980		Wilf et al.	5/4/99	
yu	AQ	5,874,960		Mairs et al.	2/23/99	
yu	AR	5,867,667		Butman et al.	2/2/99	
yu	AS	5,870,605		Bracho et al.	2/9/99	
yu	AT	5,867,660		Schmidt et al.	2/2/99	
yu	AU	5,864,711		Mairs et al.	1/26/99	
yu	AV	5,802,285		Hirviniemi	9/1/98	
yu	AW	5,799,016		Onweller	8/25/98	
yu	AX	5,790,553		Deaton, Jr. et al.	8/4/98	
yu	AY	5,790,548		Sistanizadeh et al.	8/4/98	
yu	AZ	5,764,756		Onweller	6/9/98	
yu	AA	5,761,428		Miller	6/2/98	

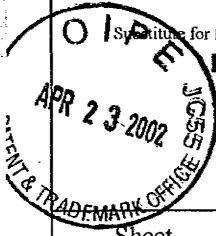
EXAMINER *[Signature]* DATE CONSIDERED 1/29/04

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

Please type a plus sign (+) inside this box → +

PTO/SB/08A

Approved for use through 10/31/99. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE



Information for form 1449A/PTO
**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**
(use as many sheets as necessary)

COMPLETE IF KNOWN

Application Number	09/629,576
Confirmation Number	
Filing Date	July 31, 2000
First Named Inventor	
Group Art Unit	ZISS
Examiner Name	WON #5
Attorney Docket No.	030048001US

Sheet 3 of 5

U.S. PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		NUMBER	Kind Code (if known)			
gju	AA	5,754,830		Butts et al.	5/19/98	
gju	AB	5,737,526		Periasamy et al.	4/7/98	
gju	AC	5,734,865		Yu	3/31/98	
gju	AD	5,732,219		Blumer et al.	3/24/98	
gju	AE	5,732,074		Spaur et al.	3/24/98	
gju	AF	5,696,903		Mahany	12/9/97	RECEIVED
gju	AG	5,673,265		Gupta et al.	9/30/97	APR 24 2002
gju	AH	5,636,371		Yu	6/3/97	Technology Center 2100
gju	AI	5,568,487		Sitbon et al.	10/22/96	
gju	AJ	5,535,199		Amri et al.	7/9/96	
gju	AK	5,426,637		Derby et al.	6/20/95	
gju	AL	5,309,437		Perlman et al.	5/3/94	
	AM					
	AN					

FOREIGN PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	Foreign Patent Document			Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		Office	Number	Kind Code (if known)				
	AO							

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

EXAMINER INITIALS	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume/issue number(s), publisher, city and/or country where published.	T
gju	AP	Murphy, Patricia, A., "The Next Generation Networking Paradigm: Producer/Consumer Model," Dedicated Systems Magazine - 2000 (pages 26-28)	
gju	AQ	The Gamer's Guide, "First-Person Shooters," October 20, 1998 (4 pages)	
gju	AR	The O'Reilly Network, "Gnutella: Alive, Well, and Changing Fast," January 25, 2001 (5 pages) http://www.open2p.com/lpt/... [Accessed 1/29/02]	

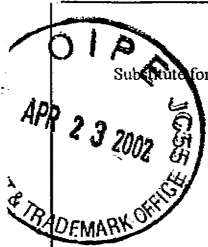
EXAMINER DATE CONSIDERED 1/29/04

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

Please type a plus sign (+) inside this box → +

PTO/SB/08A

Approved for use through 10/31/99. OMB 0651-0031
 Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE



Substitute for form 1449A/PTO

**INFORMATION DISCLOSURE
 STATEMENT BY APPLICANT**

(use as many sheets as necessary)

COMPLETE IF KNOWN

Application Number	09/629,576
Confirmation Number	
Filing Date	July 31, 2000
First Named Inventor	Virgil E. Bourassa
Group Art Unit	2155
Examiner Name	WON #5
Attorney Docket No.	030048001US

Sheet 4 of 5

U.S. PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		NUMBER	Kind Code (if known)			
	AA					

FOREIGN PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	Foreign Patent Document			Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		Office	Number	Kind Code (if known)				
	AB							

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

EXAMINER INITIALS	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume/issue number(s), publisher, city and/or country where published.	T
yw	AC	Oram, Andy, "Gnutella and Freenet Represents True Technological Innovation," May 12, 2000 (7 pages) The O'Reilly Network http://www.oreillynet.com/lpt... [Accessed 1/29/02]	
yw	AD	Internetworking Technologies Handbook, Chapter 43 (pages 43-1 - 43-16)	
yw	AE	Oram, Andy, "Peer-to-Peer Makes the Internet Interesting Again," September 22, 2000 (7 pages) The O'Reilly Network http://linux.oreillynet.com/lpt... [Accessed 1/29/02]	
yw	AF	Monte, Richard, "The Random Walk for Dummies," MIT Undergraduate Journal of Mathematics (pages 143-148)	
yw	AG	Srinivasan, R., "XDR: External Data Representation Standard," Sun Microsystems, August 1995 (20 pages) Internet RFC/STD/FYI/BCP Archives http://www.faqs.org/rfcs/rfc1832.html [Accessed 1/29/02]	
yw	AH	A Databeam Corporate White Paper, "A Primer on the T.120 Series Standards," Copyright 1995 (pages 1-16)	
yw	AI	Kessler, Gary, C., "An Overview of TCP/IP Protocols and the Internet," April 23, 1999 (23 pages) Hill Associates, Inc. http://www.hill.com/library/publications/t... [Accessed 1/29/02]	
yw	AJ	Bondy, J.A. and Murty, U.S.R., "Graph Theory with Applications," Chapters 1-3 (pages 1-47), 1976 American Elsevier Publishing Co., Inc., New York, New York	

RECEIVED
 APR 24 2002
 Technology Center 2100

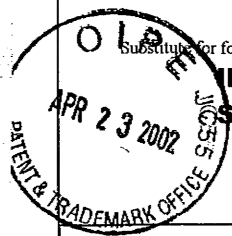
EXAMINER DATE CONSIDERED 1-29-04

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

Please type a plus sign (+) inside this box → +

PTO/SB/08A

Appr. for use through 10/31/99. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE



Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (use as many sheets as necessary)				COMPLETE IF KNOWN	
				Application Number	09/629,576
				Confirmation Number	
				Filing Date	July 31, 2000
				First Named Inventor	Virgil E. Bourassa
				Group Art Unit	2155
Examiner Name	WON				
Sheet	5	of	5	Attorney Docket No.	030048001US

U.S. PATENT DOCUMENTS					#5	
EXAMINER INITIALS	Cite No.	U.S. Patent Document NUMBER	Kind Code (if known)	Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	AA					
	AB					

FOREIGN PATENT DOCUMENTS								
EXAMINER INITIALS	Cite No.	Foreign Patent Document Office	Number	Kind Code (if known)	Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
	AC							
	AD							

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS			
EXAMINER INITIALS	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume/issue number(s), publisher, city and/or country where published.	T
<i>yu</i>	AE	Cormen, Thomas H. et al., Introduction to Algorithms, Chapter 5.3 (pages 84-91), Chapter 12 (pages 218-243), Chapter 13 (page 245), 1990, The MIT Press, Cambridge, Massachusetts, McGraw-Hill Book Company, New York	
<i>yu</i>	AF	The Common Object Request Broker: Architecture and Specification, Revision 2.6, December 2001, Chapter 12 (pages 12-1 - 12-10), Chapter 13 (pages 13-1 - 13-56), Chapter 16 (pages 16-1 - 16-26), Chapter 18 (pages 18-1 - 18-52), Chapter 20 (pages 20-1 - 20-22)	
<i>yu</i>	AG	The University of Warwick, Computer Science Open Days, "Demonstration on the Problems of Distributed Systems," http://www.dcs.warwick.ac.u... [Accessed 1/29/02]	
	AH		
	AI		
	AJ		
	AK		
	AL		

RECEIVED
 APR 24 2002
 Technology Center 2100

EXAMINER <i>George A. H.</i>	DATE CONSIDERED 1-29-04
------------------------------	----------------------------

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

COPY OF PAPERS
ORIGINALLY FILED

Attorney Docket No. 030048001US

2731
2664

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, Washington, D.C., 20231, on:

Date: 6/13/02

By: Jeanne Connelly
Jeanne Connelly



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT

JCB

ART UNIT No. 2731

RECEIVED

JUN 24 2002

Technology Center 2600

RECEIVED

JUN 27 2002

Technology Center 2100

REAPPLICATION OF:
FRED B. HOLT AND VIRGIL E. BOURASSA
APPLICATION No.: 09/629,576
FILED: JULY 31, 2000
FOR: BROADCASTING NETWORK

**Supplemental Information Disclosure Statement Within Three Months of
Application Filing or Before First Action - 37 CFR 1.97(b)**

Commissioner for Patents
Washington, D.C. 20231

Sir:

1. Timing of Submission

This information disclosure is being filed within three months of the filing date of this application or date of entry into the national stage of an international application or before the mailing date of a first Office action on the merits, whichever occurs last [37 CFR 1.97(b)]. The references listed on the enclosed Form PTO/SB/08A (modified) may be material to the examination of this application; the Examiner is requested to make them of record in the application.

2. Cited Information

- Copies of the following references are enclosed:
 - All cited references
 - References marked by asterisks
 - The following:
- Copies of the following references can be found in parent application Ser. No.
 - All cited references
 - References marked by asterisks
 - The following:
- The following references are not in English. For each such reference, the undersigned has enclosed (i) a translation of the reference; (ii) a copy of a communication from a foreign patent office or International Searching Authority citing the reference, (iii) a copy of a reference which appears to be an English-language counterpart, or (iv) an English-language abstract for the reference prepared by a third party. Applicant has not verified that the

translation, English-language counterpart or third-party abstract is an accurate representation of the teachings of the non-English reference, though, and reserves the right to demonstrate otherwise.

- All cited references
- References marked by ampersands
- The following:

3. Effect of Information Disclosure Statement (37 CFR 1.97(h))

This Information Disclosure Statement is not to be construed as a representation that: (i) a search has been made; (ii) additional information material to the examination of this application does not exist; (iii) the information, protocols, results and the like reported by third parties are accurate or enabling; or (iv) the cited information is, or is considered to be, material to patentability. In addition, applicant does not admit that any enclosed item of information constitutes prior art to the subject invention and specifically reserves the right to demonstrate that any such reference is not prior art.

4. Fee Payment

No fees are believed due. However, should the Commissioner determine that fees are due in order for this Information Disclosure Statement to be considered, the Commissioner is hereby authorized to charge such fees to Deposit Account No. 50-0665.

5. Patent Term Adjustment (37 CFR 1.704(d))

- The undersigned states that each item of information submitted herewith was cited in a communication from a foreign patent office in a counterpart application and that this communication was not received by any individual designated in 37 C.F.R. § 1.56(c) more than thirty days prior to the filing of this statement. 37 C.F.R. § 1.704(d).

Respectfully submitted,
Perkins Coie LLP



Maurice J. Pirio

Registration No. 33,273

Date: 6/13/02

Correspondence Address:

Customer No. 25096
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
Phone: (206) 583-8888

Please type a plus sign (+) inside this box →

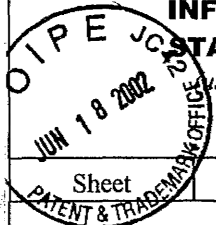
PTO/SB/08A

Approved for use through 10/31/99. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

COPY OF PAPERS
ORIGINALLY FILED

COMPLETE IF KNOWN

Substitute for form 1449A/PTO



**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**

(Use as many sheets as necessary)

Application Number	09/629,576
Confirmation Number	
Filing Date	July 31, 2000
First Named Inventor	Fred B. Holt
Group Art Unit	2731-2155 # 6
Examiner Name	WON
Attorney Docket No.	030048001US

RECEIVED

JUN 24 2002

Technology Center 260C

Sheet 1 of 1

U.S. PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		NUMBER	Kind Code (if known)			
you	AA	4,912,656		Cain et al.	3/27/90	
you	AB	5,056,085		Vu	10/8/91	
	AC					
	AD					
	AE					
	AF					
	AG					
	AH					
	AI					
	AJ					

RECEIVED

JUN 27 2002

Technology Center 2100

FOREIGN PATENT DOCUMENTS

EXAMINER INITIALS	Cite No.	Foreign Patent Document			Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		Office	Number	Kind Code (if known)				
	AK							
	AL							
	AM							
	AN							
	AO							

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

EXAMINER INITIALS	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume/issue number(s), publisher, city and/or country where published.	T
you	AP	Alagar, S. and Venkatesan, S., "Reliable Broadcast in Mobile Wireless Networks," Department of Computer Science, University of Texas at Dallas, Military Communications Conference, 1995, MILCOM '95 Conference Record, IEEE San Diego, California, November 5-8, 1995 (pages 236-240)	
you	AQ	International Search Report for The Boeing Company, International Patent Application No. PCT/US01/24240, June 5, 2002 (7 pages)	
	AR		

EXAMINER *you*

DATE CONSIDERED *1-29-04*

* EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant(s).

RELIABLE BROADCAST IN MOBILE WIRELESS NETWORKS¹

S. Alagar and S. Venkatesan
Department of Computer Science, University of Texas at Dallas
Richardson, TX 75083

J. R. Cleveland
C3 Systems Division, Electrospace Systems, Inc., A Chrysler Company
Richardson, TX 75081

ABSTRACT

This paper presents preliminary results of our research on wireless networking that supports reliable communications between nomadic hosts engaged in distributed computing and collaborative conferencing. Our network model consists of a set of low-power, radio frequency (RF) transceivers which move relative to each other across an irregular terrain subject to RF propagation impairments. The low transmitter power defines a radio coverage which limits the probability of intercept and the number of neighbors but optimizes frequency reuse. The combination of low power and propagation environment produces a network characterized by stochastic link failures. The rapidity of these failures and perturbations to the network topology defeats the use of routing policies based on maintaining routing tables or determining least cost paths. With these conditions as the background, our work addresses the need to provide reliable information exchange, mitigate bottlenecks, avoid excessive traffic, and offer scalable services without the benefit of static base station or fixed backbone support. Meeting such challenges demands a robust, flexible information transport system that delivers all required information for diverse operational scenarios. The approach emphasizes the importance of achieving guaranteed delivery across a network of limited size operating in a hostile environment rather than obtaining a high throughput per unit area, typical of commercial enterprises.

The basic premise of the protocol is that host mobility and terrain prevents *a priori* knowledge of any host location and optimum path. Message broadcasting, or flood routing, provides the means for reliable delivery of information in the presence of uncertain connectivity and node locations. Knowledge of the network results, instead, from a measure of transmitted and received message traffic. Central to the protocol is the provision for each mobile host to retain a *HISTORY* of messages broadcast to and received from its neighbor(s). A host which receives a message broadcasts an

acknowledgment to the sender, updates its local *HISTORY*, and then retransmits the message if it is not a duplicate message. Duplicated messages are discarded. If a sending host does not receive an acknowledgment from a neighbor within a certain time, it times out and resends the message. If a host does not receive an acknowledgment after several retries, it assumes that the link disconnection is not transient and stops sending the message. When a host detects a new neighbor, a handshake procedure results in the exchange of active messages not common to the respective *HISTORY* of each host. Once the handshake procedure terminates, the contents of the *HISTORY* for each host are identical. Thus, using handshake procedures, mobile hosts receive messages that they did not receive previously due to link disconnections. Idle hosts will periodically broadcast a sounding message to maintain their network presence.

1. **INTRODUCTION.** Winning the information war with complete and up-to-date intelligence is vital to the entire spectrum of possible operational requirements, whether engaged in war or corporate strategic planning. Military commanders engaged in rapid force projection, as well as public safety officers, medical staff, and corporate managers, demand accurate information regardless of location or situation. Each requires a clear and accurate picture of a changing situation to reach well-informed decisions and successful conclusion. Information must flow throughout the network toward the users at each level of the management hierarchy whether at the sustaining base or at the forward most part of the mission. Participating staff must have the capability to acquire or send accurate information that defines their space and situation. The information transport network must extend reliable voice, data, video, and imagery transmissions to nomadic users at any location. The availability of assured communications directly relates to mission success through computing, conferencing, and synchronized tasking while fixed or on-the-move. Invariably, this critical information is required when communications services normally provided by a reliable,

¹This research was supported in part by NSF under Grant No. CRR-9110177, by the Texas Advanced Technology Program under Grant No. 9741-036, and by a grant from Electrospace Systems, Inc., a Chrysler Company.

fixed infrastructure are unavailable or severely degraded.

The wireless networking of mobile (i.e., nomadic) subscribers is an emerging paradigm in the field of distributed command, control, and computing, with the potential to improve command and control responsiveness. In our context, the phrase "mobile wireless network" means that the network does not contain any static support stations. This network model supports the needs of subscribers to mobile command posts and mobile satellite ground entry points. In this role, the network must provide reliable information transfer, mitigate bottlenecks, avoid excessive traffic, offer scalable services, and, above all, adapt to dynamic topologies. The RF coverage area should conform as closely as possible to the area over which subscribers move, thereby offering a low probability of detection and improving frequency re-use. Low-cost implementation and operation are critical.

Nomadic wireless networks currently are characterized by limited bandwidth and frequent changes in link connectivity. The challenge is to allow updates from multiple users simultaneously, but only for those users that require the service. The major requirements include the capability to: minimize updates, provide fault-tolerant service, provide service scalability, and minimize communication and computation overhead. Many of the algorithms that assume static hosts or well-defined point-to-point links cannot be directly used for mobile systems due to the changes in physical connectivity and limited bandwidth of the wireless links. This has spawned considerable research in mobile computing: designing communication protocols [1, 2, 3, 4], file system operations [5, 6], managing data efficiently [7, 8], and providing fault tolerance [9]. Most research on these topics is based on a model in which the mobile hosts are supported by static base stations such as cellular telephony or personal communications systems (PCS). A typical PCS topology takes the form of a single-hop network in which each host is within radio range of the base station or all other hosts. In this paper, we consider the problem of providing reliable broadcast in mobile wireless networks where single-hop and known topologies may not exist. Applications include disaster relief operations, highly mobile military or law enforcement operations, and rapid response contingency operations where it is not economical to place support stations.

2. WIRELESS NETWORK MODEL. The model of the mobile wireless network consists of several mobile hosts distributed over an irregular terrain (Figure 1). The mobile hosts use low-power transmitters and novel, efficient receivers [10] to communicate. Emerging technologies and products for PCS applications that operate in the ISM bands with a transmitter power of 1W [11] provide the basis for practical

implementation. In this network concept, the *cell* of a mobile host is the geographical area within which the mobile hosts can directly communicate with other mobile hosts. Note that the cell of a host does not remain fixed, but moves with each attached host. The nominal cell size (R) is determined by a path loss model that denotes the *local average* received signal power relative to the transmit power. A general path loss (PL) model that has been demonstrated through measurement uses a parameter $2 \leq \mu \leq 5$ [12] to denote the power law relationship between distance and received power. Based on both analysis and measured results, $\mu \approx 4$ for the microcell propagation environment beyond a characteristic distance R_0 . The power law model takes the form [13]:

$$PL(R) = PL(R_0) + 10\mu \log(R/R_0) + X_0 \quad (1)$$

where $PL(R_0)$ gives the power loss at the characteristic distance R_0 and X_0 denotes a zero mean Gaussian random variable that reflects the fluctuations in average received power. Nelson and Kleinrock [14] show that for a slotted ALOHA network protocol, the optimum throughput occurs with a cell size defined by a range that includes an average of six nearest neighbors. Their results are reproduced in Figure 2. These results assume a random distribution of nodes across the terrain and a perfect capture condition. Perfect capture occurs when a node will always receive and detect the strongest of several simultaneous transmissions within its hearing range. The weaker signals appear as noise to the detection process. A non-capture condition occurs if simultaneous transmissions always result in collisions. The reduced power supports frequency reuse, as well as low probability of detection. Their analysis also shows that mobile hosts with sufficient

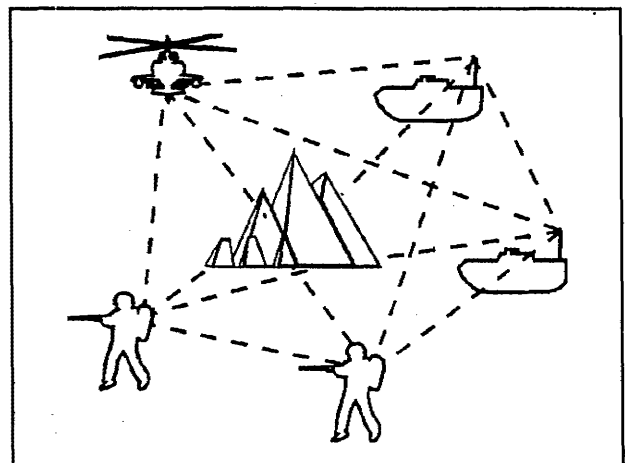


Figure 1. Model of a wireless computer network that experiences link failures due to range limitations and terrain impairments.

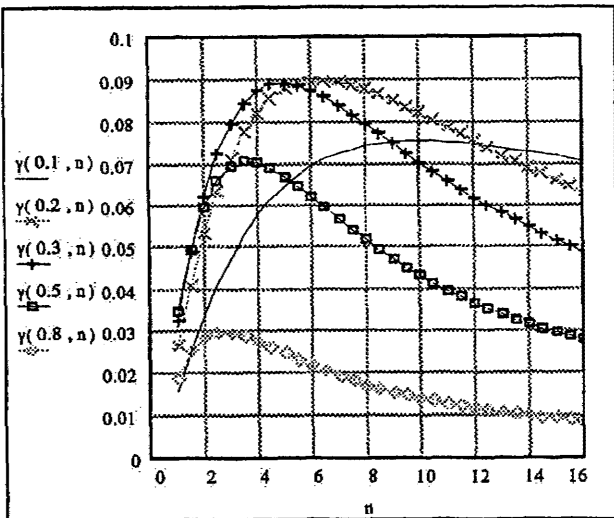


Figure 2. Normalized throughput for perfect capture with the slotted ALOHA protocol versus the average number of nearest neighbors for different probabilities that a node is busy.

transmitter power to reach all other hosts (i.e., a single-hop network) support a significantly lower throughput.

Detecting Neighbors. Neighbors are detected by a strategy common to a general class of survivable and adaptive network protocols that use sounding procedures [15, 16]. Two mobile hosts are *neighbors* if they can "hear" each other. Each host detects its neighbors by periodically broadcasting a probe

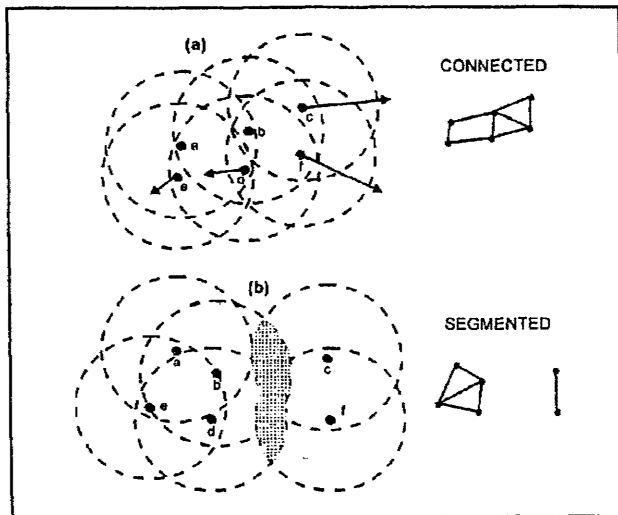


Figure 3. Examples of (a) a fully connected mobile wireless network and (b) a decomposed network due to mobility of hosts *c* and *f*. The shaded region indicates the common area within the range of hosts *c* and *f* and the rest of the network.

message. A host that hears a probe message sends an acknowledgment to the probing host. Every host maintains a list of neighbors and periodically updates the list based on acknowledgments received. When two hosts become neighbors, a wireless link is established between them, and they execute a *handshake* procedure. As part of the handshake procedure, they each update their list of neighbors.

Link Disconnections. The wireless link between two neighbors is unreliable due to RF propagation effects such as loss of line-of-sight (LOS), moving out of range, multipath fading, or inclement weather. There are two types of link disconnections: (1) transient and (2) permanent. In the transient case, a host is unable to communicate briefly with a neighbor due to: (a) the neighbor moving out of sight; (b) multipath fading; or (c) inclement weather. Multipath fading has a time dependence that varies from microseconds to seconds, depending on the terrain and the host velocity [17]. Fade depths range up to 20 dB. The stochastic behavior of such transient link disconnections is very similar to that encountered for high frequency radio networks [18]. In the permanent case, a host is unable to communicate with a neighbor because their separation exceeds the range described by the cell geometry. We assume that each mobile host can communicate with an arbitrary mobile host in its cell without any interference (from other mobile hosts in the same cell) using techniques such as TDMA or code division multiple access (CDMA) spread spectrum signaling. One such technique is presented in [16].

3. BROADCAST CONSIDERATIONS. Terrestrial networks provide the means to manage RF spectrum utilization to minimize the inherent latency for transmission, the probability of detection, and the cost of utilization not afforded by satellite services. Reliable broadcast in a mobile wireless network is not easy due to the following reasons: (1) It may be difficult to maintain a convenient structure (spanning tree, virtual ring) for broadcasting because of the mobility of hosts and the absence of an established backbone network. While an adaptive algorithm can be used to maintain the structure, the small cell size leads to cases where two neighbors may frequently move out of each others' range leading to the invocation of the adaptive algorithm frequently. (2) The wireless network itself may not be connected always (see Figure 3). They may be decomposed into several connected components for a while and merge after some time (we assume "quite often"). We also assume that permanent disconnections do not occur. In the next section we present a preliminary solution for reliable broadcast in mobile wireless network. Our solution is based on simple (restricted) flooding and handshaking.

Flooding ultimately involves transmitting the message to every

node in the network, which is a disadvantage, particularly for large networks. The main advantage of flooding is that there is little explicit overhead and network management. As a consequence, no provisions are made to store or maintain routing or management data. Instead, hosts keep track of individual messages received and determine whether or not to retransmit the message. It is well suited to network requirements for highly mobile user groups on the digitized battlefield or in disaster relief operations where there is a need for reliable delivery in the presence of uncertain connectivity and rapid topology changes [19].

4. BROADCAST PROTOCOL. To broadcast a message, a mobile host transmits the message to all of its neighbors. On receiving a broadcast message, an intermediate mobile host retransmits the message to all of its neighbors. The technique would suffice if the network remained connected forever. Additional steps are necessary to cope with network link disconnections.

For example, mobile host h_i maintains a sequence counter, CNT_i . At any instant, the value of CNT_i denotes the number of messages broadcast by host h_i . CNT_i is incremented when h_i is about to broadcast a new message. In addition, host h_i maintains a history of messages ($HISTORY_i$) it has broadcast as well as received from other hosts. The j^{th} component of $HISTORY_i$ contains information about messages broadcast from h_i and received from h_j . A rebroadcast count and a time stamp provide the means to limit the propagation of the message in a large network.

A sample pictorial representation of $HISTORY_i$ is shown in Figure 4. Host h_j has received messages 1 and 4, but not

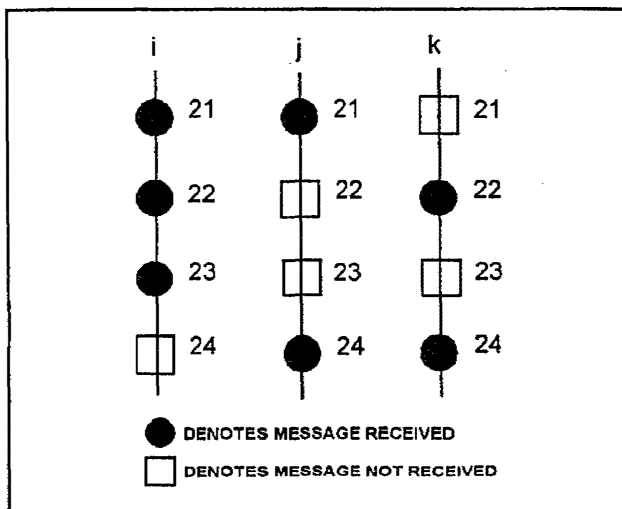


Figure 4. A snapshot of the status of the $HISTORY_i$ showing messages stored in h_i , h_j , and h_k .

messages 2 and 3. Similarly, h_k has received messages 2 and 4 from h_k , but not messages 1 and 3. It is easy to maintain $HISTORY_i$ for each h_i as an array of lists. We now describe our solution for reliable broadcast.

Normal Operation. Let us assume that host h_i wants to broadcast message m . The following occurs:

- Host h_i first increments CNT_i and then transmits message (m, h_i, CNT_i) to all neighbors. It also stores (m, h_i, CNT_i) in a buffer locally.
- When host h_j receives message (m, h_i, CNT_i) , it sends an acknowledgment to the sender, updates the i^{th} component of $HISTORY_j$ and buffers the message locally. Host h_j then retransmits the message (m, h_i, CNT_i) to its neighbors.
- If h_j receives another copy of (m, h_i, CNT_i) , it discards the message, but sends an acknowledgment to the sender.
- A mobile host h , after sending a message (m, h_i, CNT_i) to its neighbors, waits for acknowledgments from all of its neighbors. If h does not receive acknowledgment from a neighbor within a certain time, h timeouts and resends the message (with a hope that link disconnection is transient). If h does not receive acknowledgment after several retries, h assumes that the link disconnection is not transient and stops sending the message.

During periods of heavy message exchange activity, this strategy substitutes for the polling or sounding procedure described in Section 2.

Handshake Procedure. When host h_j detects a new neighbor, h_k , a handshake procedure is executed by hosts h_j and h_k :

- h_j sends $HISTORY_j$ to h_k and receives $HISTORY_k$ from h_k .
- h_j compares $HISTORY_k$ with $HISTORY_j$ to identify messages available in $HISTORY_k$ but not in $HISTORY_j$, and broadcasts those messages. Host h_k does likewise.
- h_k then receives the "new" messages sent by h_j and updates $HISTORY_k$. h_j does the same for messages received from h_k . h_j also sends these "new" messages to other neighbors.

At the conclusion of the handshake procedure, the contents of $HISTORY_j$ and $HISTORY_k$ are equal. Thus, using handshake procedure, mobile hosts receive messages that they did not receive due to link disconnections. The size of $HISTORY$ stored at each h can be reduced as follows. If the first message received by h_j from h_k is $CNT_k=t$ then it is sufficient for the k^{th} component of $HISTORY_j$ to start from t . Storage for entry of messages 1 to $t-1$ need not be provided. Further optimization can be done by storing either the $HISTORY$ of messages received or the $HISTORY$ messages not received depending on which list is smaller.

5. CONCLUSIONS AND FUTURE WORK. We have presented a protocol model designed to achieve assured delivery of information in a multi-hop nomadic wireless network. To mitigate a handicap of flood routing, the protocol includes a mechanism to restrict the retransmission of messages. The protocol accounts for the temporary separation of a node, or node segments, from other network members. Our continued research is devoted to methods which improve the protocol efficiency given the limitation of flood routing. In order to reduce the size of the buffer at each mobile host, a buffered message can be deleted after it is received by all the hosts. For each message a host receives, the host sends an acknowledgment to the sender of the message. Once acknowledgments from all hosts have reached the originator, the originator can direct the hosts to delete the message from the buffer [2]. To this end, we may have to broadcast and buffer acknowledgments also, which will increase the overhead. One of our objectives is to design an efficient acknowledgment policy that does not adversely increase the congestion and storage required at each host. Another option in deleting the buffered messages is to use timeouts, but this may not be suitable in critical applications where messages cannot be lost. Also the timeout period has to be chosen carefully (incorporating the mobility and link disconnections) so that the probability of message loss is very low. Some related research issues are: (1) deriving the necessary conditions, with respect to the host mobility pattern for our protocol to work; (2) identifying structures that are easy to maintain and are suitable for broadcasting; and (3) designing efficient routing schemes for unicasting messages.

We are investigating the efficiency and performance characteristics of survivable and adaptive network protocols with computer simulation techniques. Preliminary results will be reported on the evaluation of the algorithm in terms of message delay and acknowledgment overhead for different network sizes and routing restrictions. Our preliminary results indicate the viability of the message management protocol for collaborative computing in a dynamic computer network topology when the reliability of information is paramount.

References

- [1] Alagar, S., and Venkatesan, S. Casual ordering in distributed mobile systems. (To appear in Proc. of Workshop on Mobile Systems and Applications, Dec., 1994.)
- [2] Badrinath, B., and Acharya, A. Delivering multicast messages in networks with mobile hosts. In *Proceedings of the 13th International Conference on Distributed Computing Systems (1993)*, IEEE, pp. 292-299.
- [3] Ioannidis, J., Duchamp, D., and Maguire, G. IP-based protocols for mobile internetworking. In *Proceedings of ACM SIGCOMM Symposium on Communication Architecture and Protocols (1991)*, pp. 235-245.
- [4] Teraoka, F., Yokote, Y., and Tokoro, M. A network architecture providing host migration transparency. In *Proceedings of ACM SIGCOMM (September 1991)*.
- [5] Kistler, J., and Satyanarayana, M. Disconnected operation in coda file system. *ACM Trans. Comput. Syst.* 10, 1 (February 1992).
- [6] Tait, C. D., and Duchamp, D. Service interface and replica management algorithm for mobile file system clients. In *Proceedings of the 1st International Conference on Parallel and Distributed Information Systems (1991)*.
- [7] Alonso, R., and Korth, H. Database system issues in nomadic computing. Technical report, MITL, December 1992.
- [8] Imielinski, T., and Badrinath, B. Data management for mobile computing. *ACM SIGMOD Record* (March 1993).
- [9] Krishna, P., Vaidya, N., and Pradhan, D. Recovery in distributed mobile environments. In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems (1993)*, pp. 83-88.
- [10] Darrell, Ash and Coon, Allan, "A Micropower SAW-Stabilized Superregenerative Data Receiver," Application Note 25, RF Monolithics, Inc., July 1992.
- [11] Padgett, J. E., Gunther, C. G., and Hattori, T. Overview of wireless personnel communications. In *IEEE Commun. Mag.*, vol 33, no. 1, pp. 28-41, January, 1995.
- [12] Sousa, E. S., Silvester, J., and Papavassiliou, T. D. Computer-aided modeling of spread spectrum packet radio networks. In *IEEE J. Selected Areas Commun.*, vol. 9, pp. 48-58, 1991.
- [13] Andersen, J. B., Rappaport, T. S., and Yoshida, S. "Propagation Measurements and Models for Wireless Communications Channels," *IEEE Commun. Mag.* vol. 33, no. 1, pp. 42-49, January, 1995.
- [14] Nelson, R. D. and Kleinrock, L. The spatial capacity of a slotted ALOHA multihop packet radio network with capture. In *IEEE Trans. Commun.*, vol. COM-32, no. 6, 1984, pp. 684-694.
- [15] FED-STD-1045, Telecommunications: HF Radio Automatic Link Establishment, from Institute for Telecommunication Sciences, National Telecommunications and Information Administration, US Dept. of Commerce, 24 Jan 1990.
- [16] Baker, D. J., Ephremides, A., and Flynn, J. A. The design and simulation of a mobile radio network with distributed control. In *IEEE Journal on Selected Areas in Communications SAC-2,1* (January 1984), 226-237.
- [17] Greenstein, L. J., et. al. Microcells in personal communications systems. In *IEEE Commun. Mag.*, vol. 30, no. 12, 76-88, Dec., 1992.
- [18] Maslin, N. M. Modeling in the hf network design process. In *Fourth International Conference on HF Radio Systems and Techniques*. (London, UK, 90-94, April, 1988).
- [19] Leiner, B. M., et. al., Issues in Packet Radio Network Design. In *Proc. IEEE* vol. 75, no. 1, pp. 6-20, January, 1987.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 030048001WO	FOR FURTHER ACTION see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.	
International application No. PCT/US 01/ 24240	International filing date (day/month/year) 31/07/2001	(Earliest) Priority Date (day/month/year) 31/07/2000
Applicant THE BOEING COMPANY		

This International Search Report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This International Search Report consists of a total of 3 sheets.

It is also accompanied by a copy of each prior art document cited in this report.

1. Basis of the report

a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing :

contained in the international application in written form.

filed together with the international application in computer readable form.

furnished subsequently to this Authority in written form.

furnished subsequently to this Authority in computer readable form.

the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished

2. **Certain claims were found unsearchable** (See Box I).

3. **Unity of invention is lacking** (see Box II).

4. With regard to the **title**,

the text is approved as submitted by the applicant.

the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

the text is approved as submitted by the applicant.

the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. The figure of the **drawings** to be published with the abstract is Figure No.

as suggested by the applicant.

because the applicant failed to suggest a figure.

because this figure better characterizes the invention.

1
 None of the figures.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/24240

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/18 H04L12/56

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, IBM-TDB, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 4 912 656 A (CAIN JOSEPH B ET AL) 27 March 1990 (1990-03-27)	1-3, 7-10,12, 19,20, 22,24-27
A	column 5, line 60 -column 7, line 20	13-18, 21,28-33
Y	US 5 056 085 A (VU THU V) 8 October 1991 (1991-10-08)	1-3, 7-10,12, 19,20, 22,24-27
	column 2, line 49 -column 3, line 12	
	-/--	

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

* Special categories of cited documents :

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- *Z* document member of the same patent family

Date of the actual completion of the international search

27 May 2002

Date of mailing of the international search report

05/06/2002

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Ströbeck, A.

INTERNATIONAL SEARCH REPORT

International Application No

PCT/US 01/24240

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>ALAGAR S ET AL: "Reliable broadcast in mobile wireless networks" MILITARY COMMUNICATIONS CONFERENCE, 1995. MILCOM '95, CONFERENCE RECORD, IEEE SAN DIEGO, CA, USA 5-8 NOV. 1995, NEW YORK, NY, USA, IEEE, US, 5 November 1995 (1995-11-05), pages 236-240, XP010153965 ISBN: 0-7803-2489-7 page 237, left-hand column, line 43 -page 239, right-hand column, last line -----</p>	1-33

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 01/24240

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4912656	A	27-03-1990	NONE
US 5056085	A	08-10-1991	NONE

PATENT COOPERATION TREATY

DOCKETED

From the INTERNATIONAL SEARCHING AUTHORITY

JUN 11 2002

PCT

To:
 PERKINS COIE LLP
 Attn. Pirio, Maurice J.
 P.O. Box 1247
 Seattle, WA 98111-1247
 UNITED STATES OF AMERICA

NOTIFICATION OF TRANSMITTAL OF
 THE INTERNATIONAL SEARCH REPORT
 OR THE DECLARATION

(PCT Rule 44.1)

Date of mailing (day/month/year) 05/06/2002	
Applicant's or agent's file reference 030048001W0	FOR FURTHER ACTION See paragraphs 1 and 4 below
International application No. PCT/US 01/ 24240	International filing date (day/month/year) 31/07/2001
Applicant THE BOEING COMPANY	

- The applicant is hereby notified that the International Search Report has been established and is transmitted herewith.
Filing of amendments and statement under Article 19:
 The applicant is entitled, if he so wishes, to amend the claims of the International Application (see Rule 46):

When? The time limit for filing such amendments is normally 2 months from the date of transmittal of the International Search Report; however, for more details, see the notes on the accompanying sheet.


Where? Directly to the International Bureau of WIPO
 34, chemin des Colombettes
 1211 Geneva 20, Switzerland
 Facsimile No.: (41-22) 740.14.35

For more detailed instructions, see the notes on the accompanying sheet.
- The applicant is hereby notified that no International Search Report will be established and that the declaration under Article 17(2)(a) to that effect is transmitted herewith.
- With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:
 - the protest together with the decision thereon has been transmitted to the international Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.
 - no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.
- 4. Further action(s):** The applicant is reminded of the following:

 Shortly after **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90bis.1 and 90bis.3, respectively, before the completion of the technical preparations for international publication.

 Within **19 months** from the priority date, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase until 30 months from the priority date (in some Offices even later).

 Within **20 months** from the priority date, the applicant must perform the prescribed acts for entry into the national phase before all designated Offices which have not been elected in the demand or in a later election within 19 months from the priority date or could not be elected because they are not bound by Chapter II.

Name and mailing address of the International Searching Authority  European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Claude Berthon
--	--

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the PCT Applicant's Guide, a publication of WIPO.

In these Notes, "Article", "Rule", and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only.

What parts of the international application may be amended?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When?

Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How?

Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Administrative Instructions, Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped), whether

- (i) the claim is unchanged;
- (ii) the claim is cancelled;
- (iii) the claim is new;
- (iv) the claim replaces one or more claims as filed;
- (v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter:

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."
2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."
3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."
4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the same time of filing the amendments with the International Bureau, also file a copy of such amendments with the International Preliminary Examining Authority (see Rule 62.2(a), first sentence).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, where upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see Volume II of the PCT Applicant's Guide.

United States Patent [19]

Cain et al.

[11] Patent Number: 4,912,656

[45] Date of Patent: Mar. 27, 1990

[54] **ADAPTIVE LINK ASSIGNMENT FOR A DYNAMIC COMMUNICATION NETWORK**

[75] Inventors: Joseph B. Cain; Stanley L. Adams, both of Indialantic; John W. Nieto; Michael D. Noakes, both of Melbourne, all of Fla.

[73] Assignee: Harris Corporation, Melbourne, Fla.

[21] Appl. No.: 249,180

[22] Filed: Sep. 26, 1988

[51] Int. Cl.⁴ G06F 11/00

[52] U.S. Cl. 364/514; 364/402; 340/827

[58] Field of Search 364/514, 200 MS File, 364/900 MS File, 402, 407; 371/11.2, 11.1, 8.1, 68.2; 340/827, 826, 825.03; 455/8

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,706,080 11/1987 Sincoskie 340/825.02
4,788,721 11/1988 Krishnan et al. 379/221

Primary Examiner—Parshotam S. Lall
Assistant Examiner—S. A. Melnick
Attorney, Agent, or Firm—Antonelli, Terry & Wands

[57] **ABSTRACT**

An adaptive link assignment scheme for dynamically changing communication node topologies such as satellite networks, fleets of ships or aircraft, etc. Periodically each node in the network transmits topology information to all the nodes in the network. Then each node determines the degree of connectivity of the network. It is preferred that the network be at least triconnected, and if the network is less than that each node determines what connections it can make to improve the network connectivity. If more than one alternative is available, a choice is made based on line of sight endurance and then on traffic delay. The identification of the selected connection is then broadcast to all the nodes in the network. Each node thus receives the proposed changes from all the network nodes, and each node then resolves conflicts between the broadcast selections and determines what change it should make. Finally, the changes are implemented. The scheme emphasizes network connectivity to bring the network to a triconnected state and then emphasizes line of sight endurance and reduction of traffic delay.

24 Claims, 7 Drawing Sheets

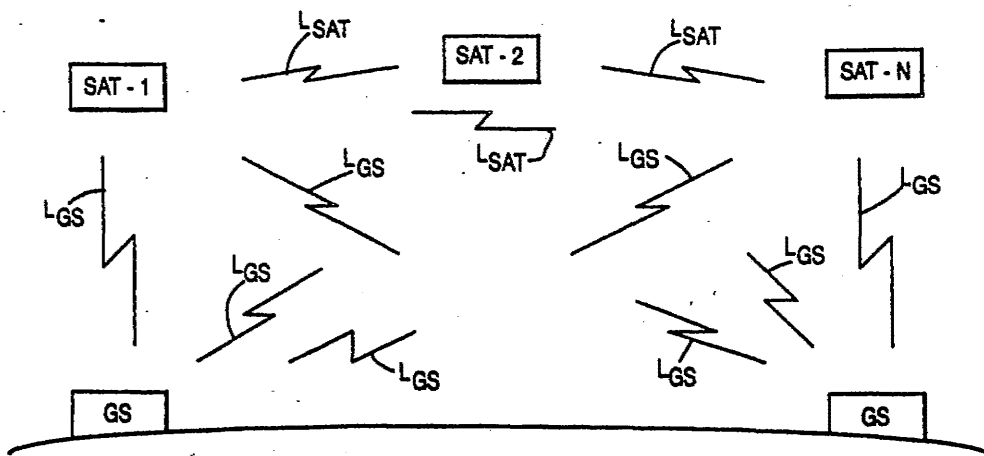


FIG. 1

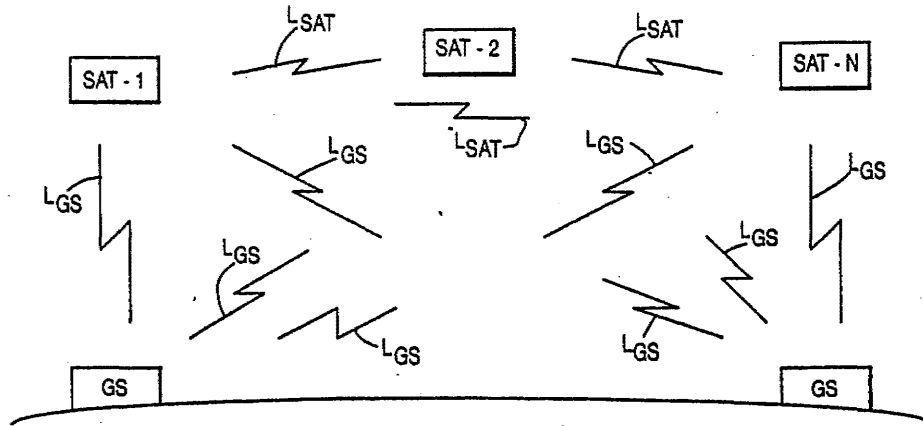
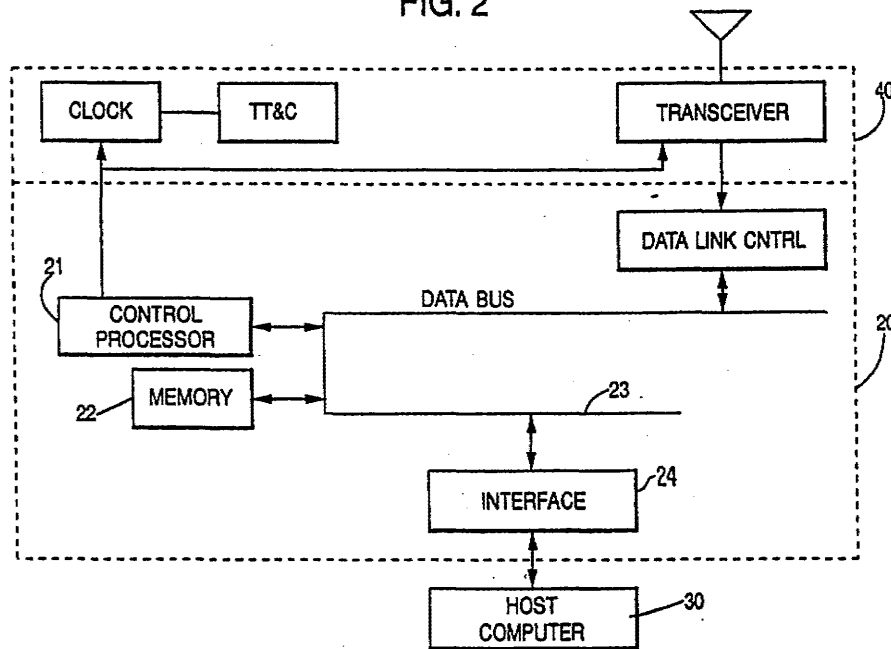


FIG. 2



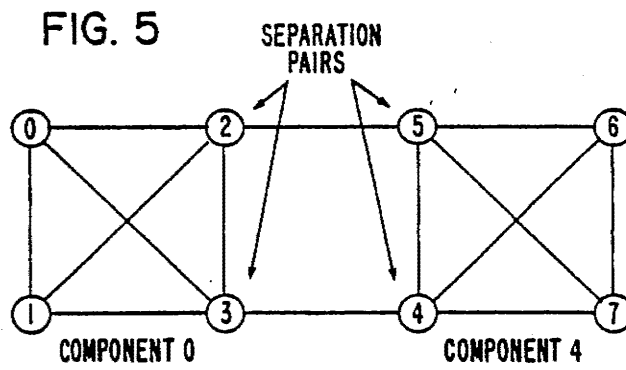
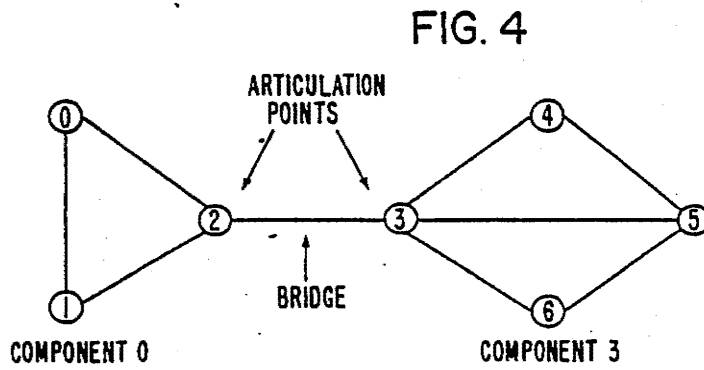
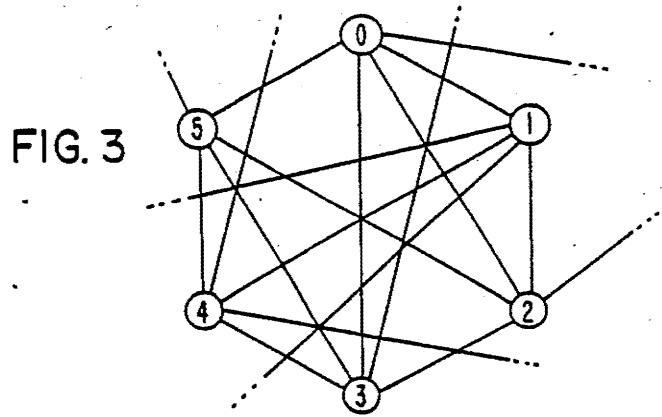


FIG. 6.1 $D_T = \sum_m D_m$ WHERE $D_m = \sum_k f_{mk} D_{mk} (f_{mk})$

FIG. 6.2 $D_{mT} = \sum_j t_m^j \sum_{\text{link \& path}} D_{\text{link}} (f_{\text{link}})$

FIG. 6.3 $M_d = \frac{D_{mT} - D_{mT}^{\text{new}}}{D_{mT}^{\text{new}}}$, WHEN $D_{mT}^{\text{new}} \leq D_{mT}$
 $M_d = \frac{D_{mT} - D_{mT}^{\text{new}}}{D_{mT}}$, WHEN $D_{mT}^{\text{new}} > D_{mT}$

FIG. 6.4 $C = n_L + N^2 n_{L-1} + N^4 n_{L-2} + \dots + N^2 (L-1) n_1$

FIG. 6.5 $M_{\text{net}} = (M_c + \eta M_d) (1 + \epsilon Q_{ij}^{\text{LOS}})$

FIG. 7

```
Program LinkAssignment
loop forever
begin
  dispatch on "Evaluate"
  EvaluateAlternatives(topology, ephemeris);
  deadline on "Broadcast"
  Broadcast;
  dispatch on "Resolve Conflicts"
  ResolveConflicts(messages, components, graphs, iteration);
  dispatch on "Initiate Link Establishment"
  InitiateLinkEstablishment(changes);
end;
```

FIG. 8

```
Procedure EvaluateAlternatives(topology, ephemeris)
Procedure Connect(connectivity, predicate, enumerate)
begin
  components := connectivity(graphs);
if not predicate(components)
begin
  selections := Presort(enumerate(self, components, graphs));
  RankSelections(selections, components);
  return_from EvaluateAlternatives;
end;
end;

graphs := Partition(RemoveImminentOutages(topology, ephemeris, self));
Connect(Identity, Connected, JoinGraphs);
Connect(Biconnectivity, Biconnected, MergeBiconnected);
Connect(Triconnectivity, Triconnected, MergeTriconnected);
Connect(Identity, False, GeneralChanges);
```

FIG. 9

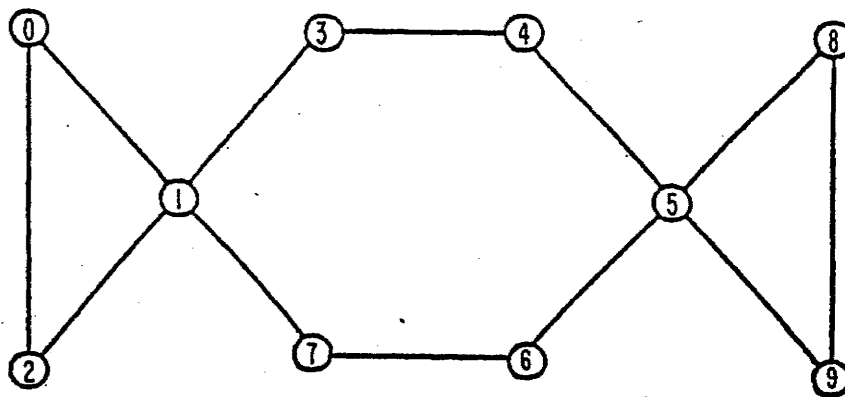


FIG. 10

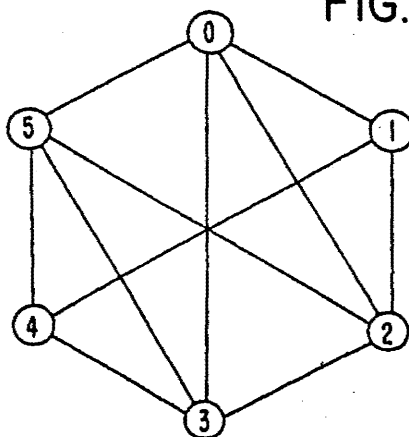


FIG. II

Procedure Resolve(messages, components, graphs, iteration)

```
changes := { };
for selection in SortSelections(selections)
  if SelectionNodesStillAvailable(selection) and
    SelectionStillMergesComponents(selection, components) and
    SelectionNotBadBreak(selection, components)
    begin
      components := AdjustComponents(selection, components);
      MarkAffectedNodes(selection);
      if ChangeAffectsMe(selection)
        changes := Adjoin(selection, changes);
    end
```

ADAPTIVE LINK ASSIGNMENT FOR A DYNAMIC COMMUNICATION NETWORK

The U.S. government has rights in the present invention under NRL Contract N00014-86C-2056.

BACKGROUND OF THE INVENTION

The present invention relates in general to communications systems and is particularly directed to an adaptive link assignment scheme for dynamically changing communication node topologies.

In multistation or multinode communication networks it is essential to maintain communication paths between nodes to assure that communication can be had between any node in the network and any other node in the network. This is particularly critical in remote communication networks such as communication satellite networks. While equipment malfunction can result in loss of a communication link, in dynamically positioned networks such as satellite networks, communication links between nodes can also be lost if the two nodes move such that they are no longer in line of sight (LOS).

Generally each node is capable of communication over several communication links, which by way of example might be laser links or radio frequency links, or a combination thereof. Thus, for example, a communication satellite might be provided with several antennas permitting it to have simultaneous communication with several other satellites over separate paths. Problems can arise in the assignment of the communication links when there are more LOS neighbors of a node than there are communication ports. In such a situation, a link assignment scheme must be utilized to determine the communication links to be established and maintained. Such a scheme must consider the desired network connectivity, determine schedules for establishing and disconnecting links, and command the establishment and disconnection of links in accordance with those schedules, all by utilizing the communication resources available to the node.

A satellite network might include, for example, 48 satellites. It is desirable that any one of the 48 satellites be able to communicate with any other of the 48 satellites. While a single communication path from any one satellite to any other satellite permits such communication, it is desirable that redundant paths be provided to assure no loss of communication in the event that a link is broken, due to either a failure at a node or a failure in a link. Thus, although the minimum link assignment that can provide the desired communication is a single path between nodes, it is desirable that at least two independent communication paths be provided between any one node and any other node in the network, and it is even more preferable that three paths exist so that communications can be maintained even in the event of breakdown of two communication paths.

Design considerations of communication paths in static networks are discussed in the text *Computer Networks*, by A. S. Tanenbaum, Prentice Hall, 1981. The paper "A Distributed Link Assignment (Reconstitution) Algorithm for Space-based SDI Networks," by J. B. Cain, S. L. Adams, M. D. Noakes, P. J. Knoke, and E. I. Althouse, published at pages 29.2.2-29.2.7 in the MILCOM '87 Conference Record, October 1987, describes an algorithm for assigning communication links in a dynamic multinode communication network. Addi-

tional detail is presented in the Harris Corporation ADNMS Technical Report "Selected Algorithm Description," dated Nov. 5, 1987 and prepared for The Naval Research Laboratory under Contract No. N00014-86-C-2056. The present invention considers key aspects of the matter and presents an improved algorithm.

SUMMARY OF THE INVENTION

The present invention is concerned with a distributed and dynamic version of such a problem. The number of distinct assignments in a network having n nodes is on the order of $(n!)$. As a consequence, an exhaustive search to find the best network topology is impractical. Instead, the algorithm operates in an incremental fashion, continually making small changes in topology to compensate for predictable link outages due to nodes losing line of sight from other nodes. Further, the network rapidly reestablishes communication in the event of massive outages. Consequently, the algorithm is capable of determining an acceptable topology by searching only a fraction of the possible topologies. Heuristic methods are utilized, rather than an exhaustive search, and less than optimum solutions are accepted so long as they meet certain minimum criteria. Survivability of the communication network and timely message delivery are essential, particularly in military applications. Thus, the number of discrete or disjoint paths between nodes and the message delay time are the principal criteria of concern. With the present invention it is possible to maintain good topology under stress and to determine link assignments with little computational time.

The present invention thus provides a network topology that is robust and that carries the desired traffic load with low delay. The algorithm emphasizes improving the robustness or number of connections of the network topology if the topology is fragile, even at the expense of increased delay. Once the topology is reasonably robust, the algorithm optimizes both delay time and connectivity. Nevertheless, it is easily possible to emphasize other properties, such as emphasizing reduction of delay at the expense of connectivity, if desired. A dynamic network requires the use of an adaptive routing algorithm so that packets in the network will still be delivered after typical changes in the links, but the link assignment algorithm does not rely on the details of such a routing scheme. An island is defined to be a set of connected nodes, i.e. a set of nodes for which there is at least one path between all pairs of nodes. It is important that all the nodes in the network be collected into a single island but it is possible that an unfortunate sequence of link outages will split the network into multiple islands. The present invention utilizes a network management scheme which includes a periodic broadcast of a node's connectivity information to all the other nodes within its island. This can be achieved robustly by the use of a constrained flooding algorithm.

Additionally, in accordance with the present invention, all nodes have synchronized timing accurate to within about 100 milliseconds to permit coordinating of certain actions, such as transmission of the periodic topology update information. Further, each node or satellite can determine the locations of all other nodes in its island in order to determine its line of sight neighbors. This position determination is aided through the periodic topology update information.

The network of nodes may become partitioned or fragmented in a stressed environment, with the result

that it consists of multiple islands with no communication between islands.

The algorithm of the present invention is able to work with any scheme that permits the acquisition of lost nodes. However it provides direct support for a specific model of island joining. This scheme supposes that every node contains L identical ports, any one of which may be used in a scanning mode to search for disconnected nodes. The search may be either directed, using ephemeris information, or undirected if this information is not available. In later sections of this disclosure, in which a specific reduction to practice is detailed by way of an example, it is assumed that a node will generally attempt to maintain one free port for this purpose and for implementing link changes. However, this is just one particular approach to node acquisition and the remainder of the algorithm is not dependent on this choice.

A goal of the algorithm is to maintain a network that is sufficiently robust, i.e. has a plurality of redundant paths between all pairs of nodes, so that it is rarely necessary to acquire lost nodes.

Data is transmitted between nodes by a packet communication technique in which discrete packets of data are transmitted from a sending node to a receiving node, perhaps passing through one or more intermediate nodes which relay the packet. In addition to data, each packet includes header information which identifies the receiving node and the packet length, among other things. Topology updating information is transmitted in separate packets which for example might be initiated by a periodic timer. The headers of these packets identify these packets as topology information rather than as data.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other aspects and advantages of the present invention are more apparent in the following detailed description and claims, particularly when considered in conjunction with the accompanying drawings. In the drawings:

FIG. 1 diagrammatically illustrates a multinode satellite communications network comprised of a plurality of communications satellites interlinked with one another and with a plurality of associated ground stations;

FIG. 2 is a block diagram illustration of the communication and digital processing equipment at an individual node, whether a ground station or a communications satellite, of the communication network of FIG. 1;

FIG. 3 is a two dimensional topology diagram of an exemplary multinode communication network;

FIGS. 4 and 5 are graphical representations of communication networks helpful in understanding of the present invention;

FIG. 6 sets forth various equations useful in an understanding of the invention;

FIGS. 7 and 8 list processing steps of heuristic procedures used in accordance with the present invention;

FIG. 9 is another graphical representation, similar to FIGS. 4 and 5;

FIG. 10 is another two dimensional topology diagram of a multinode communication network; and

FIG. 11 lists a further processing step in accordance with the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention resides primarily in the adaptive link assignment system and process for the member nodes of a multinode communication network employing conventional communications circuits and components, and not in the particular detailed configurations thereof. Accordingly, the structure, control and arrangement of conventional circuits and components are illustrated in the drawings by readily understood block diagrams which show only those specific details that are pertinent to the present invention, thus not obscuring the disclosure of the present invention with structural details which are readily apparent to those skilled in the art having the benefit of the description herein. Thus, the block diagram of illustrations of the figures do not necessarily represent the mechanical structural arrangements of the exemplary system but are primarily intended to illustrate the major structural components of this system in a convenient functional grouping, whereby the present invention may be more readily understood.

To provide an illustrative example in the description to follow, the communications environment to which the present invention is applied is described as a satellite communications system comprised of a plurality of communications satellites interlinked with one another and with a plurality of associated ground stations. However, it should be realized that the invention is not limited to use with satellite communications or with this particular network, but is applicable to any multinode network in which communications between source and destination nodes may take place over multiple paths through the nodes of the network. It should also be noted that the network topology can be either dynamic, changing with time, or static, effectively time invariant. The present invention is particularly advantageous when incorporated in the communications network where the network topology is dynamic, for example with multiple moving vehicles such as aircraft, spacecraft, ships, or land vehicles, and where the various communications stations or nodes, the distances between nodes, and the communications paths between nodes may vary with time, consequently requiring real time effective tracking of changes in the network topology.

In the satellite communications network diagrammatically illustrated in FIG. 1, a plurality of geographically distributed terrestrial or ground stations GS communicate with one another by means of a multiplicity of communications satellites SAT via respective ground to satellite or satellite to ground communications paths Lgs and satellite to satellite communication paths Lsat. Ground stations GS and satellites SAT correspond to the network nodes which are linked to one another via transmission paths Lgs and Lsat. Each node, whether a ground station GS or satellites SAT, contains conventional transceiver, tracking, acquisition, and signal processing equipment, as shown in block diagram form in FIG. 2.

The typical node depicted in FIG. 2 includes a packet switch 20 which has a communications bus 23 that communicates through an interface unit 24 to a host computer 30. Computer 30 executes the various user processes and is the source or destination of all user data. Packet switch 20 sends data packets from host computer 30 to other nodes, delivers data packets to com-

puter 30, or relays packets to other nodes. Packet communication is typically done with use of a set of layered protocols similar to the seven layer International Standards Organization Reference Model of Open System Interconnection. In the node of FIG. 2, data link control is a low level function of layer 2 of the International Standards Organization Reference Model and provides error free transmission of packets. The data link control may be implemented using a standard synchronous data link communication protocol. For a more detailed explanation of this seven layer model, as well as a number of standard protocols that may be employed in accordance with the present invention, attention is directed to the above-mentioned book *Computer Networks*.

Control processor 21 within packet switch 20 provides basic control of the packet and may be a general purpose computer. Control processor 21 is primarily responsible for executing the network layer, or layer 3, protocols. The network layer is a layer in which the link assignments are determined. The adaptive link assignment algorithm of the present invention is executed by control processor 21.

Associated with control processor 21 is a random access memory 22 in which the packet buffers, control software, and link assignment tables reside. Memory 22 provides temporary storage for packets while control processor 21 is deciding whether to transmit the packet to another node, to deliver the packet to host computer 30, or to relay a received packet to another node. Such decision is based upon the address in the packet header. The packet routes available to other nodes depend upon link assignments which are updated periodically.

FIG. 2 also shows conventional satellite communications equipment which makes up the physical layer, or layer 1, of the node, in the form of a transceiver unit 40 which contains transmitter, receiver, modulator and demodulator units, tracking telemetry and control units, and a clock. These units interface with the packet switch through conventional data link control components. The details of the configuration and operation of such equipment are unnecessary for an understanding of the present invention, and so such details are not set out herein. For further information with regard to such, attention is directed to standard communication texts such as *Digital Communications by Satellite*, by J. J. Spilker, Prentice Hall, 1977.

Carrier and bit synchronization may be accomplished using conventional mechanisms, such as described in the article "Carrier and Bit Synchronization in Data Communication — A Tutorial Review," by L. E. Franks, *IEEE Transactions on Communications*, Vol. COM-28, August 1980, pages 1107-1121. Precise timing of node operations may be implemented using techniques such as described in the articles "Network Timing/Synchronization For Defense Communications," by H. A. Stover, pages 1234-1244 and "Synchronization of A Digital Network," by J. Heymen, et al., pages 1285-1290, *IEEE Transactions on Communications*, Vol. COM-28, August 1980.

FIG. 3 illustrates a network having 6 nodes, designated 0-5. Any network or island is named in accordance with the lowest node number within that network or island. Thus, the network of FIG. 3 can be named network 0. Each node within network 0 of FIG. 3 has a plurality of communication ports, illustratively depicted in FIG. 3 as five ports for each node and so is capable of simultaneous communication over a plurality of communication links. Thus, for example, node 0 has

a communication link established with each of node 1, node 2, node 3, and node 5, and has an additional communication port available for use in a search mode. Similarly, nodes 2, 3, and 5 of FIG. 3 each have four established communication ports and one port used in a search mode. Each of nodes 1 and 4 has three established communication links and one port used in a search mode, leaving each of these two nodes with a free port.

The degree of connectivity or robustness of a graph topology is determined by the number of disjoint paths between all pairs of nodes in the graph. There are two forms of disjointness. The less stringent of the two is link disjointness. Two paths from node A to node B are link disjoint if they do not have a common link. Two paths are node-disjoint if they do not have a node in common. Two paths that are node disjoint are clearly also link disjoint. The major concern during each topology update is whether line-of-sight outages are about to split the network into disjoint islands. In general, the network is sufficiently robust that this is extremely unlikely. However, if this situation has been detected, all the nodes take whatever action may be necessary to prevent the separation. Two link-disjoint paths between all pairs of nodes in a graph are required to guarantee that no single path failure will separate the graph into islands. Similarly, two node-disjoint paths between all pairs of nodes are required to guarantee that no single node failure will separate the graph into two islands.

A graph is biconnected if there are at least two node-disjoint paths between every pair of nodes. Similarly, a graph is triconnected if there are at least three node-disjoint paths between every pair of nodes.

Important features of a non-biconnected graph are the sets of biconnected nodes, the articulation points (or separation points), and the bridges. The nodes of a non-biconnected graph may be collected into biconnected components. The successive merging of these components creates a biconnected graph. FIG. 4 shows a non-biconnected graph with two biconnected components, named component 0 and component 3. The bridge between biconnected components and the articulation points at each end of the bridge represent the critical link and nodes, respectively, any of which will separate the graph into two or more components if removed.

A simple algorithm is set forth in "Depth-first Search and Linear Graph Algorithms" by R. Tarjan, published in *SIAM J. Computing*, Vol. 1, June 1972, pp. 146-160, the disclosure of which is incorporated herein by reference. This algorithm uses a depth-first search through a graph to find sets of biconnected components, sets of articulation points, and sets of bridges of a connected graph, doing so in on the order of $(n+e)$ time, where n is the number of vertices and e is the number of edges. This information is required to improve the robustness of the graph without a lengthy search process. From this information it can be seen that the graph of FIG. 4 can be made biconnected by adding a link from any node in component 0 to any node in component 3, except for the articulation points. Thus, for example, a link from node 0 to node 4 will make the entire graph biconnected. In accordance with the present invention, a high priority is assigned to creating at least a biconnected network using this algorithm to direct the critical additions.

Once this has been done a considerably more complicated algorithm is used in a similar fashion to produce a triconnected network. The algorithm is discussed in

"Dividing a Graph into Triconnected Components," by J. E. Hopcroft and R. E. Tarjan, published in *SIAM J. Computing*, Vol. 2, September 1973, pp. 135-158, the disclosure of which is incorporated herein by reference. This algorithm also uses a depth-first search and runs in on the order of $(n+e)$ time. The algorithm finds triconnected components and separation pairs. A separation pair is a pair of nodes which, if removed from the graph, will partition the graph into two or more components. Thus, the graph of FIG. 5 is biconnected, but it has triconnected components. In this example the separation pairs are (2, 3) and (4, 5), and the triconnected components are (0, 1, 2, 3) and (4, 5, 6, 7). These algorithms can be used to rapidly improve a graph's connectivity; however, it appears to be impractical to generalize this approach to higher levels of connectivity. Beyond this level of connectivity more compute intensive algorithms are necessary in order to improve network robustness, and, generally, a triconnected network is considered adequately robust.

If more than one new link is available to provide an improved level of connectivity, various "tie breakers" are available. The following discussion will focus on two particular metrics that will tend to lead to improved network topologies. However, the invention disclosed does not depend on the precise formulation of these heuristics. Other metrics that capture important characteristics of a network may be accommodated.

In any network, some potential paths will have LOS visibility for a longer period of time than others. It is important that the topology have enduring connectivity, and so links with a long-term LOS visibility are more valuable than links with only short-term visibility. Let $T_{LOS_{ij}}$ be the length of time that nodes i and j will have LOS visibility. This number can range from 0 to ∞ depending on orbital parameters of the nodes. In order to establish a new link, there is an acquisition overhead (which is a function of the link technology) that can help to define a lower limit $T_{LOS_{min}}$ for LOS visibility below which it is not useful to establish a new link. Likewise, there is an upper limit $T_{LOS_{max}}$ beyond which all LOS endurance can be considered equivalent. One way in which the quality of the endurance of the LOS visibility can be rated is the following link durability measure, having four categories:

Q LOS	Endurance		
1	T_{min}^{LOS}	$< T_{ij}^{LOS}$	$< T_{max/16}^{LOS}$
2	$T_{max/16}^{LOS}$	$< T_{ij}^{LOS}$	$< T_{max/8}^{LOS}$
3	$T_{max/8}^{LOS}$	$< T_{ij}^{LOS}$	$< T_{max}^{LOS}$
4	T_{ij}^{LOS}	$< T_{max}^{LOS}$	

Any potential link $(i-j)$ having $T_{LOS_{ij}} < T_{LOS_{min}}$ is automatically rejected. This quality measure can be used as a tie-breaker for competing connections that might otherwise provide equivalent connectivity.

The above link connectivity metric is based entirely on the distribution of node-disjoint paths. While this permits the construction of a network that is tolerant of failures, it is clear that the link assignment algorithm should also make changes to respond to dynamic changes in traffic requirements. A delay metric can be combined with the connectivity metric to create an overall topology metric by which to judge the network.

A representative delay metric may be computed using only traffic data available at each node. In particular, the calculation at each node is based solely on the traffic sourced at that node and on traffic forwarded or

relayed through the node. Thus, the node considers the effect on delay of all traffic transmitted by it. This allows a node to determine the approximate effect of link assignment decisions made by it on average packet delay.

The link metric to be used for link (m,k) is denoted by $D_{mk}^{(f_{mk})}$. This value reflects propagation, transmission, and queuing delays.

The link assignment algorithm is used with an adaptive routing algorithm in order to provide robust data delivery. Adaptation of routing tables is done in response to changes in traffic loads and topological changes. The link assignment algorithm does not attempt to respond as quickly to changes in traffic patterns as does the adaptive routing algorithm, because modifications to the topology are more difficult than changes to routing.

The goal is to minimize as closely as possible the total delay in the network, i.e. to minimize the value D_T as computed by the equation of FIG. 6.1. However, a node cannot do this directly and exactly because the node does not have the global information required to compute D_m at other nodes in the island. Instead, the effects on the network of flow transmitted from the node are directly modelled. Any node m evaluates its impact on D_T by constructing a minimum cost spanning tree from node m .

Let f_{ik} be the long term average flow on link (i,k) , and let v_m be the total long term average traffic at node m for destination node j , including both the traffic sourced at node m and the traffic relayed or forwarded by it. The shortest path spanning tree from node m is generated using an algorithm set forth in "A Note on Two Problems in Connection with Graphs," by E. Dijkstra, *Numer. Math.*, Vol. 1, pp. 269-271, 1959. Then the value D_{mT} is computed using the equation of FIG. 6.2 which weights the routing tree by the amount of traffic T_{jm} at node m for all destinations. Note that by including T_{jm} as a weighing term, this metric favors topologies in which there are direct connections from node m to the destinations with the largest amount of traffic. After computing this metric for the current topology, D_{mT}^{new} is found using new minimum cost spanning trees. The improvement is given by:

$$\Delta D_{mT} = D_{mT} - D_{mT}^{new}$$

If ΔD_{mT} is positive, then the change reduces the delay, and the ability of the network to carry the traffic is improved. When a link is added from node m to node j , it must be given a fictitious traffic flow for the purposes of this calculation. The link utilization is set equal to the average link utilization on the other links from node m .

When the network is relatively weakly connected, the connectivity algorithms direct the changes to improve connectivity, with tie-breakers based on the LOS durability and the delay metric. The potential connection with the highest link durability metric, $Q_{LOS_{ij}}$, is chosen. In case there are multiple link in the same class, the link with the best delay metric improvement is selected. This strategy emphasizes improving connectivity so long as the network is less than tri-connected.

Once the island becomes triconnected, the algorithm gives more emphasis to improving delay. This is done with a combined metric that reflects both delay and connectivity. The connectivity metric used is the distribution of link-disjoint paths. Because of the difference in units of the two metrics, they are normalized to pro-

vide the fractional change prior to combination. The overall strategy is, thus, that when the network is single connected, emphasis is given to establishing new paths that make the network biconnected, with LOS endurance and delay used as tie breakers; when the network is biconnected, emphasis is given to making the network triconnected, with LOS endurance and delay again used as tie breakers; and when the network is triconnected, emphasis is given equally to improving distribution, LOS endurance, and delay.

When the network is triconnected, the metrics can be combined by summing the normalized fractional improvement of each metric. The fractional change in the delay metric is defined by the equations of FIG. 6.3.

The connectivity metric computes the number of pairs of nodes with single link-disjoint paths, the number with double link-disjoint paths, the number with triple link-disjoint paths and so on. This information is used to form an integer. Where there are L ports at each node, N node pairs used to compute the metric, and n_j node pairs with j link-disjoint paths, the integer is calculated with the formula of FIG. 6.4. The choice of N^2 , N^4 , etc. for coefficients provides a large change in C if the connectivity changes from $(L-j)$ -connected to $(L+1-j)$ -connected. Smaller values of C are preferred, just as for delay D . The number of disjoint paths between all pairs of nodes may be computed using one of the max-flow min-cut algorithms such as the Dinic algorithm as set forth in the text *Computer Networks* by A. S. Tanenbaum. Prentice Hall, 1981, pp 44 to 52, the disclosure of which is incorporated herein by reference.

As before, let C be the connectivity of the baseline topology and let C_{new} be the connectivity of the modified network. The normalized fractional change is, then:

$$M_c = \frac{C - C_{new}}{NC}, \text{ when } C_{new} \leq C, \text{ and}$$

$$M_c = \frac{C_{new} - C}{NC}, \text{ when } C_{new} > C.$$

The N in the denominator scales the calculation appropriately relative to delay. Improving the network from being $(d-1)$ connected to being d connected is worth about a factor of 2 in delay.

A change breaking a link and adding a new link is accepted if the total change in the value M_{net} , as computed by the equation of FIG. 6.5, exceeds some threshold Δ . The parameter η allows changes in the emphasis between optimizing connectivity and optimizing delay. The parameter ϵ allows emphasizing the link durability. For changes that are simply link additions without any breaks, changes are accepted with a lower tolerance.

The adaptive link assignment algorithm consists of two decoupled activities, namely node acquisition and improving connectivity, that run in parallel. In the following discussion, several of the top-level routines are presented in a pseudo-Pascal syntax. Declarations have been removed and artificial control structures have been introduced in the interests of conciseness. These routines refer to a number of global variables that capture the state of the link assignment algorithm.

A major goal of the algorithm is to collect all the nodes into a single graph. This may be done using a search link, as previously discussed. When a node is acquired, the link that located it is assigned to make the link connection. A new search link can be established

using the algorithm described below. A unique number in the range 0 to $N-1$, where N is the number of nodes in the island, is assigned to every node. This value is used in conjunction with an iteration counter to assign rotating priorities to each node. A small amount of arbitration must occur to reassign node numbers, align iteration counters, and so on.

The connectivity improvement algorithm is run once per topology update cycle. It consists of four loosely coupled subframes: Evaluate Alternatives, Broadcast Selections, Resolve Conflict, and Initiate Link Establishment, as shown in FIG. 7. "Dispatch on" indicates that the arrival of the specified timer signal should cause the associated procedure to run. During the Evaluate Alternatives subframe, each node analyzes the current topology and determines what changes it should make to improve the island's metric. These changes are broadcast to all of the nodes in the island in the following subframe. This may occur as soon as the selections have been determined, but it must happen by the set deadline. This is indicated by Deadline On "Broadcast." The selections are developed in an incremental fashion to provide the best estimate for a set of offers to be broadcast in the event evaluation must be halted prematurely. It is likely that a number of the selections will compete for the same resource. These problems are detected and resolved during the Resolve Conflicts subframe. Finally, the process of acquiring any new paths is initiated in the Initiate Link Establishment subframe. This may take several iterations.

There is only one message transaction per iteration. This is important since propagation delays may represent a significant portion of the total frame time. This feature is supported by the complete topology information available at every node and the use of deterministic conflict resolution algorithms. Knowledge of the topology, the transmitted selections, and the current priorities enables every node to correctly anticipate the final actions of every other node without a separate acknowledge cycle. This technique is thus called "implicit acknowledgement."

The organization is sufficiently flexible to permit a range of implementation approaches within each subframe. In particular, the Evaluate Alternatives subframe can be almost arbitrarily complex. In a simple implementation, every node offers to make the "obvious" change to respond to problems in the graph. For example, if predictable LOS failures threaten to split the graph into two disjoint components, then every node in each component would attempt to add a link to a node in the other component. Of course, it is only necessary to add one link in order to ensure that the graph remains connected and just two links to make it biconnected. Thus, in the conflict resolution stage, most of the offers are disregarded, resulting in lost potential improvements. One can employ more sophisticated algorithms incorporating lookahead to anticipate which links should be utilized to connect the components and then reanalyze the graph on the assumption that these links will be provided.

The Evaluate Alternatives subframe is used to perform the required analysis of the current graph. The action of each node is determined by its resources and the configuration of the current graph. A node may have a number of spare ports, it may be using all its ports except for the search port, or even its search port may be in use. In a similar way, the graph is in one of

four easily distinguished states; it may be in immediate risk of splintering, it may be less than biconnected, less than triconnected, or it may be triconnected. As discussed above, every node knows which nodes have links to each other, which are in LOS, and for how much longer.

If the graph is in danger of fracturing and a node does not have a search port, the node attempts to break one of its links and considers offering it to every node in every other island. If the node has a search port then it considers adding a link to every node in every island. If it has one or more free ports in addition to the search port, and there are multiple islands, a very rare case, it will be willing to add several links in a single iteration if no other nodes are in a position to provide necessary links.

If the graph is connected but is less than biconnected then nodes without a search port are permitted to attempt to find an acceptable deletion to reestablish this port. However such a deletion will not be approved if it creates additional biconnected components or, worse yet, splits the island. Nodes with just a search port offer to use it to merge components but such offers will be accorded a low priority. Finally, nodes with free ports attempt to improve the connectivity.

If the graph is biconnected or triconnected then nodes without a search port attempt to obtain one as before, nodes with a search port consider rearrangements of their links, and nodes with free ports try to make useful additions. If a node is rearranging its links, then the network metric must be considerably improved to justify the disruption. If the graph is not triconnected then the additions will seek to merge the components; otherwise more general changes are considered.

Nodes without a search port attempt to restore the port by finding a deletion that does not damage the graph too seriously. If there are several possible deletions, then the least damaging one is selected. Such a node normally marks the link as though it is about to lose line-of-sight and then waits several iterations before severing the link. This permits the adaptive routing algorithm, for example, to make adjustments in the routing table to eliminate traffic from this path prior to disconnecting it. An exception to this occurs when it is necessary to make an immediate change to avoid graph fragmentation. If this node's link is the most desirable offer to avoid fracturing, then the path is deleted immediately.

If all of the ports except the search port are in use, the node generally considers rearrangements of its primary links to improve the graph metric. Such a change is accepted only if it provides a significant improvement in the metric. In these cases, the search port is used to make the addition and the link to be deleted is marked as though it were losing LOS. However if the graph is not biconnected, the search port is offered to improve the connectivity. It will be recovered during later iterations.

Finally nodes with free ports, in addition to the search port, attempt to add a port as required. Although many nodes may offer to make a change to solve a particular problem, only one of the solutions is selected after the Evaluate Alternative subframe.

The algorithm strikes a balance between making changes in a timely manner and making so many alterations in a single iteration that the metric evaluations that justify a change become meaningless. This is done

by generally limiting each node to making just one change per iteration. However, because of potential conflicts between decisions, a node may issue several offers each iteration. These offers are broadcast to all the nodes in the graph, and the best changes are selected. Each offer states the problem it is addressing and the associated metric. The best changes are selected by sorting the offers and removing redundant ones.

There are four levels of connectivity that can be easily determined using the current algorithms: imminent loss of connectivity, connected, biconnected, and triconnected. The goal of the algorithm is to improve the connectivity by one level per iteration until the graph is triconnected and then to make general changes to adapt to fluctuating traffic requirements.

With the exception of one special case, a node investigates just those changes in which it will be directly involved and that require one addition and/or deletion. In cases in which there are many such alternatives, simple criteria are used to form a preliminary ordering of the choices, and then the network metric is used to incrementally establish a more accurate ordering. During this analysis the node does not consider how its changes will interact with decisions being made by other nodes. It is possible that an apparently desirable change will not seem so attractive once some other selection is implemented. This is inevitable since performing an accurate analysis of the interactions might require a substantial amount of lookahead and backtracking, and this accounts for the reluctance to make too many actions during a single iteration.

To clarify the issues presented here, FIG. 8 presents the overall flow chart for control in the Evaluate Alternative subframe. Note that this is merely one implementation of the scheme described within this disclosure. Four classes of actions are considered during every iteration: determining whether the graph is in danger of splintering, adding paths to merge biconnected components, adding paths to merge triconnected components, and general changes for well connected graphs.

This procedure uses a programming style, namely procedural abstraction, that is not commonly encountered in Pascal programs and which may appear a little confusing at first. It is observed that the flow of control is essentially the same at all four levels of connectivity and that the only differences are in the procedures used to analyze the topology and to determine whether it is appropriate to take a particular class of action, and to determine the manner in which the potential changes are found. This common pattern of usage is captured within the subprocedure "Connect()", set out in FIG. 8. The three arguments to this procedure are themselves procedures. The first argument performs the required connectivity analysis, the second one tests the current connectivity, and the final one generates the set of possible changes. This set of changes is sorted once using coarse criteria and is then sorted more carefully using the network metric. An invocation of "Connect()" is quite easily understood if one mentally substitutes the names of the procedures for their corresponding use within the procedure. Recall that control may return from "EvaluateAlternatives()" after the completion of the required work or when the allotted time interval is exceeded.

The main result of this procedure, a set of selections, is returned in the external variable "selections." Two further values are computed that are used in later subframes. These are stored in the external variables

"graphs" and "components." The first contains the set of connected graphs and the second stores the components at the current level of interest. The procedure "Identity()" simply returns its single argument, a set of connected graphs, while "Biconnectivity()" and "Triconnectivity()" return the biconnected components and triconnected components of the graph respectively. "Connected()" returns true if there is a single connected graph. "Biconnected()" and "Triconnected()" return true if the network is biconnected or triconnected respectively, and "False()" always returns false. Here it is assumed that case matters in identifier names. The remaining four functions generate sets of possible changes and are discussed below.

The procedure "RemoveImminentOutages()" uses the topology and ephemeris information to create a representation of the current network. Links that are expecting line-of-sight outages are not included. The variable "self" is passed by reference so that it can be assigned to the data structure that represents the current node. This set of nodes is partitioned into a set of one or more disjoint graphs, and the result is stored in the external variable "graphs." The procedure "Connect()" is then called to examine the four levels of connectivity.

"JoinGraphs()" enumerates the allowable selections. In this case it has been determined that the network is in danger of fragmenting, and so each node considers additions to every node in every disjoint graph. If the node does not have a spare port, it attempts to break one of its links, subject to the constraint that a deletion will not create a new biconnected component within its subgraph.

If the graph is not at risk of fragmentation but is not biconnected, then links must be added between pairs of components to remove bridges. The importance of this action is sufficient to justify nodes with just their search port free using that for a connection. However, nodes with all their ports in use are not required to break a path. Instead, those nodes calmly seek a deletion that restores the search port. As before, a deletion is not considered if it creates a new biconnected or less component. These offers are analyzed during the Resolve Conflicts subframe, and the best set is chosen.

When there are multiple components, there may be preferred components to merge. In FIG. 9 it can be seen that nodes in component 0 would prefer to add a link to nodes in component 5, as this will remove two bridges in one step, while nodes in component 1 will equally prefer components 0 and 5. This situation presents several opportunities for added sophistication. Since it is preferable for a link to be added between components 0 and 5, there is little to be gained by having nodes in component 1 make an offer. In fact, given sufficient computing resources it is possible to determine exactly which pair of nodes in components 0 and 5 should join and then to make decisions moving directly towards triconnectivity during the current iteration. This desire is motivated by the recognition that in the example just presented it would be inadvisable to do more than add a single link without additional analysis.

If the graph is at least biconnected, the triconnected components algorithm is executed to determine whether it is also triconnected. This algorithm is more complicated than the previous one but its asymptotic time complexity is also in the order of $(n+e)$. If there are triconnected components then "MergeTricomponents()" is called to compute the set of possible changes. This function is much like "MergeBicomponents()"

except that nodes with only the search port free must find an acceptable replacement port before volunteering the search port. Such rearrangements require that the new metric be substantially better than the current metric.

Once the network is at least triconnected, the combined metric is used to direct changes through the procedure "GeneralChanges()." Nodes with free ports attempt to add links to other nodes with spare ports, nodes with just the search link free consider rearrangements of their primary links, and those without a search port attempt to free one of their ports.

If a node has free ports but cannot find any additions to make, then the graph may have settled into a local maxima with respect to connectivity. Ignoring dynamic changes, it is necessary to provide deletions and additions among distant nodes in order to improve the connectivity further. Consider the graph of FIG. 10 in which every node has four primary ports and all nodes are in LOS, and the graph is triconnected. It can be seen that nodes 1 and 4 are using only three of their ports but that they are already connected to each other. The graph can only become fully connected by permitting a node to command a deletion between two other nodes. For example, if the link between 0 and 3 is broken, then it becomes possible to obtain a maximally connected graph by adding links between 0 and 4 and between 1 and 3. In general, nodes are not permitted to exert this much control over the graph, as it is too expensive to evaluate all the possible combinations. However, it is very easy to recognize this special case when it does happen and then to take appropriate action. A node attempting this change commands the deletion and both additions. It is possible that other nodes have useful changes to make even if the current node requires a change of this sort to fully utilize its ports.

The possible changes are resorted in "Presort()" (FIG. 8) to enhance the likelihood that the best solutions will be found in the time permitted. A selection cannot be transmitted based only on the initial ranking; it must have been analyzed formally to be a candidate for broadcasting. In this step changes that are likely to balance the number of free ports at each node are preferred, then those that result in persistent links, and finally those that connect the node to those neighbors for which the greatest amount of traffic is being either generated or forwarded.

Once the possible changes have been presorted, the network metric is used in "RankSelections()" to refine the ordering. The initial ranking is used to ensure that the most sensible changes will be evaluated formally before the time for this slot is overrun. However, a selection cannot be transmitted unless it has been validated by this routine. In general, there may be several sets of equally important changes to make; e.g., several biconnected components to be merged. Metrics are evaluated for each set in round robin fashion computing a metric for one selection in each set, inserting it in the appropriate position, and proceeding to the next set. This is continued until all the selections in every set have been sorted.

The Broadcast selections subframe is used to transmit the selections to all the other nodes in the island. This is done using a flood mechanism to ensure reliable delivery. A message must indicate the type of change being commanded and the source of the message. It may specify a deletion, up to two additions, and a network metric.

A key aspect of the new algorithm is the elimination of a separate acknowledge phase. Since every node receives the same topology messages and set of selections they can all deterministically arrive at the same conclusions regarding which changes should be approved. Every selection includes a network metric that indicates the level of connectivity achieved by the change, the relative importance of the type of change (e.g. break vs add/break), a measure of the connectivity, the longevity of a link addition, and the delay metric. These last three measures are combined in different ways depending on the current level of connectivity. To ensure that competing selections can be resolved correctly it is necessary to include additional information to guarantee that they are labeled uniquely. Every node is assigned a rotating priority that is included in each transmitted packet. Finally a counter is included to distinguish the selections from a given node still further.

Resolving conflicts is relatively simple and is shown in FIG. 11. The received selections are sorted according to the network metric included in each selection and are then processed in order. In general a node may make at most one addition and/or one deletion. As such selection is accepted, the affected nodes are marked appropriately. If the selection makes an addition and the graph is not triconnected or if the selection makes a deletion then the components are recomputed. Typically there will be many selections that attempt to achieve the same effect. For example two biconnected components may be merged by a single link addition and all other offers should be discarded. Before a selection is accepted it is determined whether it attempts to join two components that have already been merged. Similarly a selection that proposes a link deletion must be checked against the database to determine whether it, in connection with other selections that have already been approved for the current iteration, will split the island or introduce new biconnected components.

The execution of `ResolveConflicts()` may determine that the current node must make one or more changes. If the change calls for a deletion then the relevant link is typically marked as though it is about to experience a conventional link outage. However if the island is in danger of fragmenting, the deletion must occur immediately.

Although the present invention has been described with reference to preferred embodiments, various changes and rearrangements could be made, and still the result would be within the scope of the invention.

What is claimed is:

1. For use in a multinode communication graph including a plurality of communication nodes and having a known graph topology with each node a part of a component, each component made up of a number of nodes between one and the full number of nodes in the graph, each node having a plurality of communication ports, each port capable of establishing a communication link with a communication port of another node, a method of assigning ports to communication links to maintain the connectivity of the graph, said method comprising within any one node the steps of:

- (a) analyzing the current graph topology to determine the current graph connectivity state;
- (b) determining the available changes said one node is capable of making to improve the graph connectivity;
- (c) broadcasting a description of the determined changes to other nodes in the graph;

- (d) receiving from other nodes in the graph broadcasts of descriptions of changes determined by the other nodes to be available to such other nodes to improve the graph connectivity;
- (e) resolving conflicts between link assignments in the determined changes of all nodes in the graph to determine a new link assignment for said one node; and
- (f) initiating establishment of a new communication path with another node in accordance with the determined new link assignment.

2. A method as claimed in claim 1 wherein the step of analyzing the current graph topology comprises the steps of:

- (i) identifying components within the graph which are at least triconnected with other components of the graph;
- (ii) identifying components within the graph which are less than triconnected with other components of the graph; and
- (iii) identifying components within the graph which are less than biconnected with other components of the graph

3. A method as claimed in claim 2 wherein the step of analyzing the current graph topology comprises the further step of:

- (iv) identifying components within the graph which are in danger of splintering from the graph due to loss of an existing communication link.

4. A method as claimed in claim 1 wherein the step of analysing the current graph topology comprises the step of identifying components within the graph which are in danger of splintering from the graph due to loss of an existing communication link.

5. A method as claimed in claim 3 or 4 wherein the step of determining the available changes comprises the step of identifying a port within said one node that can be utilized to establish a new communication link with one of the components identified as in danger of splintering.

6. A method as claimed in claim 5 wherein the step of identifying a port comprises the step of evaluating existing communication links from said one node to identify an existing communication link that can be broken to provide a port without creating a new biconnected or less component.

7. A method as claimed in claim 2 wherein the step of determining the available changes comprises the step of identifying a port within said one node that can be utilized to establish a new communication link with a node of another component with which the component of said one node is less than biconnected.

8. A method as claimed in claim 2 or 7 wherein the step of determining the available changes comprises the step of identifying a port within said one node that can be utilized to establish a new communication link with a node of another component with which the component of said one node is less than triconnected.

9. A method as claimed in claim 7 wherein the step of identifying a port includes the steps of determining the line of sight endurance of all port within said one node that are available to establish the new communication link; and identifying the one of said all ports having the greatest line of sight endurance.

10. A method as claimed in claim 9 wherein the step of identifying a port includes the further step of determining the effect on delay of all traffic transmitted by

17

said one node that would result from establishing the new communication link with the identified port.

11. A method as claimed in claim 7 wherein the step of identifying a port includes the step of determining the effect on delay of all traffic transmitted by said one node that would result from establishing the new communication link with the identified port.

12. A method as claimed in claim 1 wherein the step of resolving conflicts includes comparing the descriptions of available changes received from other nodes with the available changes of said one node; identifying commonly available changes; and determining the new link assignment from the commonly available changes.

13. In a multinode communication graph including a plurality of communication nodes and having a known graph topology with each node a part of a component, each component made up of a number of nodes between one and the full number of nodes in the graph, each node having a plurality of communication ports, each port capable of establishing a communication link with a communication port of another node, an arrangement for assigning ports to communication links to maintain the connectivity of the graph, said arrangement comprising within each node;

- (a) means for analyzing the current graph topology to determine the current graph connectivity state;
- (b) means for determining the available changes said one node is capable of making to improve the graph connectivity;
- (c) means for broadcasting a description of the determined changes to other nodes in the graph;
- (d) means for receiving from other nodes in the graph broadcasts of descriptions of changes determined by the other nodes to be available to such other nodes to improve the graph connectivity;
- (e) means for resolving conflicts between link assignments in the determined changes of all nodes in the graph to determine a new link assignment for said one node; and
- (f) means for initiating establishment of a new communication path with another node in accordance with the determined new link assignment.

14. An arrangement as claimed in claim 13 wherein said means for analyzing the current graph topology comprises:

- (i) means for identifying components within the graph which are at least triconnected with other components of the graph;
- (ii) means for identifying components within the graph which are less than triconnected with other components of the graph; and
- (iii) means for identifying components within the graph which are less than biconnected with other components of the graph.

15. An arrangement as claimed in claim 14 wherein said means for analysing the current graph topology comprises:

- (iv) means for identifying components within the graph which are in danger of splintering from the

18

graph due to loss of an existing communication link.

16. An arrangement as claimed in claim 13 wherein said means for analysing the current graph topology comprises means for identifying components within the graph which are in danger of splintering from the graph due to loss of an existing communication link.

17. An arrangement as claimed in claim 15 or 16 wherein said means for determining the available changes comprises means for identifying a port within said one node that can be utilized to establish a new communication link with one of the components identified as in danger of splintering.

18. An arrangement as claimed in claim 17 wherein said means for identifying a port comprises means for evaluating existing communication links from said one node to identify an existing communication link that can be broken to provide a port without creating a new biconnected or less component.

19. An arrangement as claimed in claim 14 wherein said means for determining the available changes comprises means for identifying a port within said one node that can be utilized to establish a new communication link with a node of another component with which the component of said one node is less than biconnected.

20. An arrangement as claimed in claim 14 or 19 wherein said means for determining the available changes comprises means for identifying a port within said one node that can be utilized to establish a new communication link with a node of another component with which the component of said one node is less than triconnected.

21. An arrangement as claimed in claim 19 wherein said means for identifying a port includes means for determining the line of sight endurance of all ports within said one node that are available to establish the new communication link; and means for identifying the one of said all ports having the greatest line of sight endurance.

22. An arrangement as claimed in claim 21 wherein said means for identifying a port further includes means for determining the effect on delay of all traffic transmitted by said one node that would result from establishing the new communication path with the identified port.

23. An arrangement as claimed in claim 19 wherein said means for identifying a port includes means for determining the effect on delay of all traffic transmitted by said one node that would result from establishing the new communication link with the identified port.

24. An arrangement as claimed in claim 13 wherein said means for resolving conflicts includes means for comparing the descriptions of available changes received from other nodes with the available changes of said one node; means for identifying commonly available changes; and means for determining the new link assignment from the commonly available changes.

* * * * *

60

65

- [54] FLOOD-AND-FORWARD ROUTING FOR BROADCAST PACKETS IN PACKET SWITCHING NETWORKS
- [75] Inventor: Thu V. Vu, West Melbourne, Fla.
- [73] Assignee: Harris Corporation, Melbourne, Fla.
- [21] Appl. No.: 391,197
- [22] Filed: Aug. 9, 1989
- [51] Int. Cl.⁵ H04Q 11/04
- [52] U.S. Cl. 370/60; 370/94.1
- [58] Field of Search 370/60, 60.1, 94.1, 370/94.2, 94.3, 85.13, 85.14

- [56] **References Cited**
- U.S. PATENT DOCUMENTS**
- 4,399,531 8/1983 Grande et al. 370/60
- 4,905,233 2/1990 Cain et al. 370/94.1

OTHER PUBLICATIONS

Computer Networks, by Andrew S. Tananbaum, Prentice Hall, Englewood Cliffs, N.J., 1981.

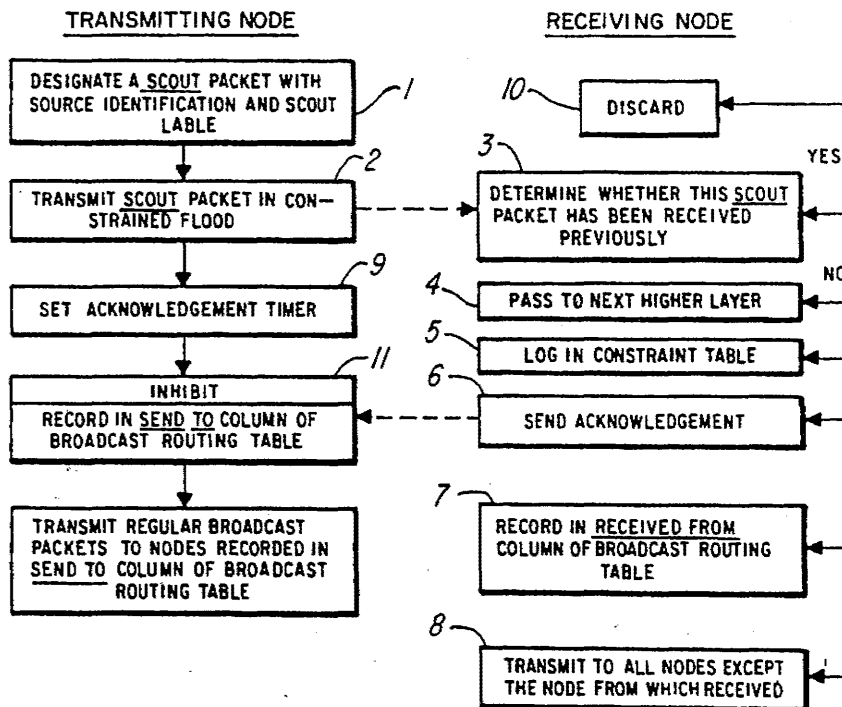
"Reverse Path Forwarding of Broadcast Packets," Y. K. Dalal and R. M. Metcalf, *Communications of the ACM*, vol. 21, pp. 1040-1048, Dec. 1978.

Primary Examiner—Douglas W. Olms
Assistant Examiner—Wellington Chin
Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus

[57] **ABSTRACT**
 A routing algorithm for broadcast packets in packet

switching networks, utilizing a "flood-and-forward" technique. In such networks, data are often transmitted in great quantities from a sensor node to all other nodes in the network, or in a subnetwork, over point-to-point links. Existing broadcast routing algorithms, including multidestination addressing, constrained flooding, minimum spanning tree forwarding, and reverse path forwarding, suffer from an excessive use of bandwidth, a poor choice of routes, or a costly need for memory or computing power. In flood-and-forward routing, periodically a data packet is designated as a Scout packet and is transmitted in a constrained flood broadcast transmission. The Scout packet is identified by a Source Id and a Scout Label. Each receiving node sends a Ack Scout packet to the node from which it first receives a particular Scout packet, acknowledging receipt of that packet. Each relaying node keeps a log of nodes from which it has received Ack Scout packets and sends subsequent, non-scout packets to those same nodes. This flood-and-forward broadcast routing algorithm thus offers the best selection of routes, as in constrained flooding, and the least consumption of bandwidth, as in minimum spanning tree forwarding, while keeping the overhead cost of storage and processing to a low level. With the support of a reliable link service, the algorithm performs well in delivering critical data to all reachable destinations despite to-be-expected losses of packets, links, or nodes.

3 Claims, 10 Drawing Sheets



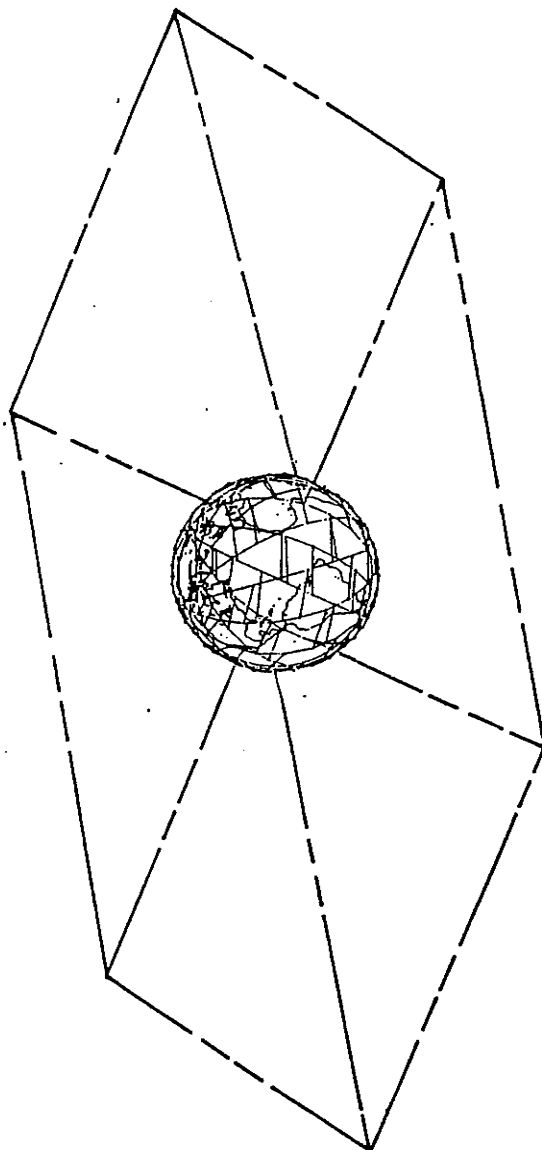


FIG. 1

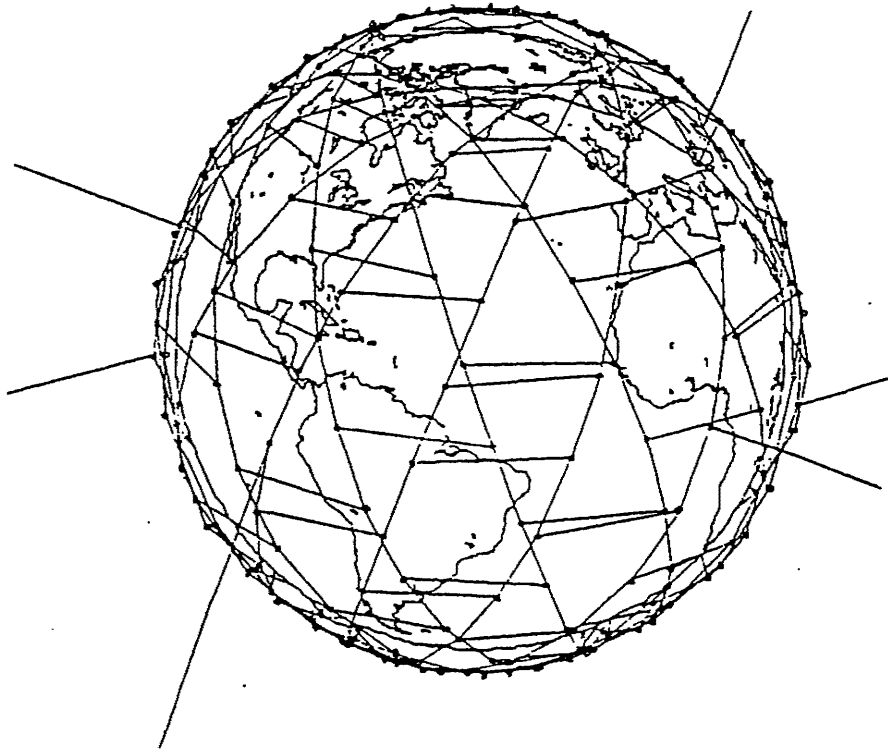


FIG. 2

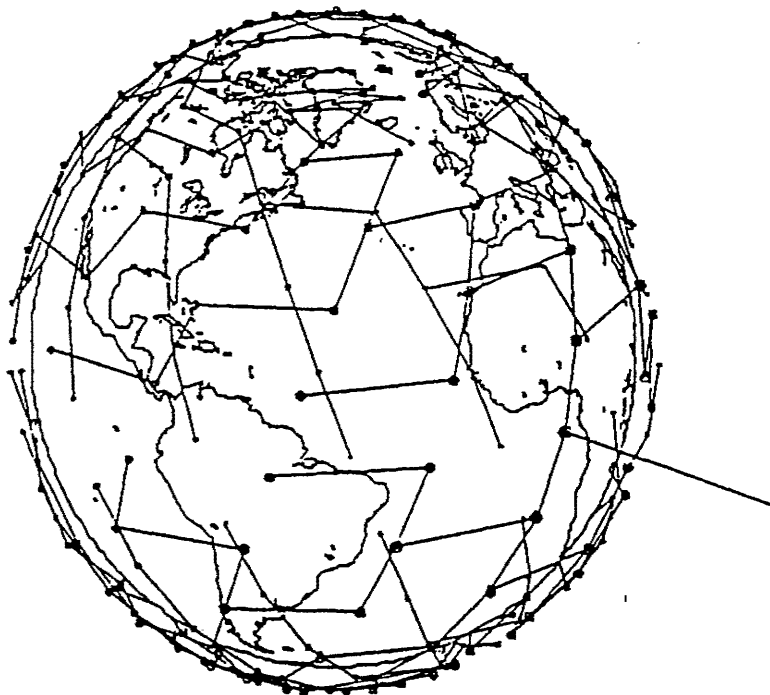


FIG. 3A

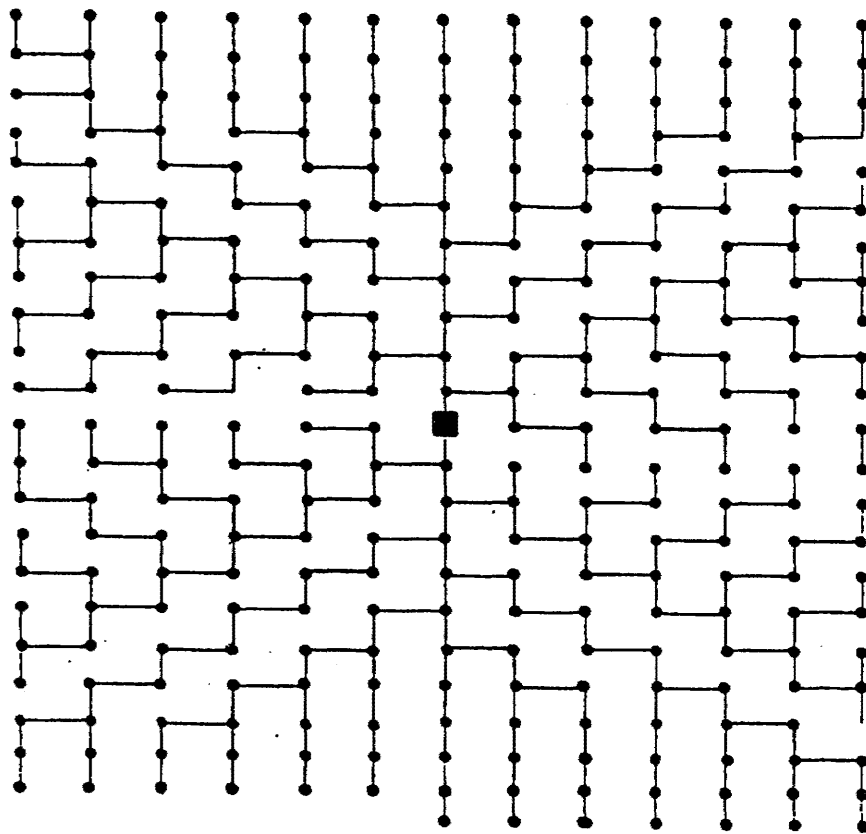


FIG. 3B

```
PROCEDURE GENERATE_BROADCAST(DATA_UNIT) IS
BEGIN
  IF (CURRENT_TIME > SCOUT_LAST_SENT_TIME + NON_FLOOD_PERIOD) THEN
    -- IT'S TIME TO SEND A SCOUT PACKET
    GENERATE_FLOOD_BROADCAST(SCOUT_LABEL, DATA_UNIT);
    SCOUT_LAST_SENT_TIME := CURRENT_TIME;
    INCREMENT_SCOUT_LABEL;
  ELSE IF (CURRENT_TIME > ROUTES_LAST_UPDATED_TIME + ROUTES_LIFE) THEN
    -- ROUTES ARE NOT UP TO DATE
    PUT_PACKETS_ON_HOLD(DATA_UNIT);
  ELSE
    -- USE BROADCAST ROUTING TABLES
    GENERATE_NON_FLOOD_BROADCAST(CURRENT_ROUTES, DATA_UNIT);
  END IF;
END GENERATE_BROADCAST;
```

FIG. 4

```
PROCEDURE PROPAGATE_FLOOD_BROADCAST(SCOUT_PACKET, LINK_ARRIVED_ON) IS
BEGIN
  NOT_YET_SEEN := CHECK_CONSTRAINT_TABLE(SCOUT_PACKET);
  IF (NOT_YET_SEEN) THEN
    ACCEPT_AND_LOG_PACKET(SCOUT_PACKET);
    -- FORWARD SCOUT PACKET
    FORWARD_LINKS := ALL_LINKS - LINK_ARRIVED_ON;
    FORWARD_PACKET(SCOUT_PACKET, FORWARD_LINKS);
    -- SET UP MECHANISM FOR EXTRACTING ROUTES FROM SCOUT PACKET
    SOURCE_ID := SCOUT_PACKET.SOURCE_ID;
    SCOUT_LABEL := SCOUT_PACKET.SCOUT_LABEL;
    ACK_SCOUT_TIMER(SOURCE_ID, SCOUT_LABEL) := CURRENT_TIME +
      ACK_SCOUT_PERIOD;
    BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).SEND_TO := NULL;
    BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).RECEIVED_FROM :=
      LINK_ARRIVED_ON;
    -- SEND ACK SCOUT PACKET
    PREPARE_ACK_SCOUT_PACKET(SOURCE_ID, SCOUT_LABEL, ACK_SCOUT_PACKET);
    FORWARD_LINKS := LINK_ARRIVED_ON;
    FORWARD_PACKET(ACK_SCOUT_PACKET, FORWARD_LINKS);
  END IF;
END PROPAGATE_FLOOD_BROADCAST;
```

FIG. 5

```
PROCEDURE GENERATE_FLOOD_BROADCAST(SCOUT_LABEL, DATA_UNIT) IS
BEGIN
  SOURCE_ID := OWN_ID;
  PREPARE_SCOUT_PACKET(SOURCE_ID, SCOUT_LABEL, DATA_UNIT, SCOUT_PACKET);
  FORWARD_LINKS := ALL_LINKS;
  FORWARD_PACKET(SCOUT_PACKET, FORWARD_LINKS);
  -- SET UP MECHANISM FOR EXTRACTING ROUTES FROM SCOUT PACKET
  ACK_SCOUT_TIMER(SOURCE_ID, SCOUT_LABEL) := CURRENT_TIME +
    ACK_SCOUT_PERIOD;
  BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).SEND_TO := NULL;
  TIME_TO_INSTALL_ROUTES := CURRENT_TIME + ACK_SCOUT_PERIOD;
  SCHEDULE_TIME_TO_INSTALL_ROUTES(SCOUT_LABEL);
END GENERATE_FLOOD_BROADCAST;
```

FIG. 6

```
PROCEDURE RECEIVE_ACK_SCOUT(ACK_SCOUT_PACKET, LINK_ARRIVED_ON) IS
BEGIN
  SOURCE_ID := ACK_SCOUT_PACKET.SOURCE_ID;
  SCOUT_LABEL := ACK_SCOUT_PACKET.SCOUT_LABEL;
  IF (CURRENT_TIME <= ACK_SCOUT_TIMER(SOURCE_ID, SCOUT_LABEL)) THEN
    BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).SEND_TO :=
      BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).SEND_TO +
      LINK_ARRIVED_ON;
  END IF;
END RECEIVE_ACK_SCOUT;
```

FIG. 7

```
PROCEDURE INSTALL_ROUTES_EVENT(SCOUT_LABEL) IS
BEGIN
  CURRENT_ROUTES := SCOUT_LABEL;
  ROUTES_LAST_UPDATED_TIME := CURRENT_TIME;
  -- BROADCAST PACKETS WHICH WERE PUT ON HOLD BECAUSE NO ROUTES
  -- WERE AVAILABLE
  WHILE (MORE_PACKETS_ON_HOLD) LOOP
    RELEASE_PACKETS_ON_HOLD(DATA_UNIT);
    GENERATE_NON_FLOOD_BROADCAST(CURRENT_ROUTES, DATA_UNIT);
  END LOOP;
END INSTALL_ROUTES_EVENT;
```

FIG. 8

```
PROCEDURE GENERATE_NON_FLOOD_BROADCAST(SCOUT_LABEL, DATA_UNIT) IS
BEGIN
  SOURCE_ID := OWN_ID;
  PREPARE_BROADCAST_PACKET(SOURCE_ID, SCOUT_LABEL, DATA_UNIT, PACKET);
  FORWARD_LINKS := BROADCAST_ROUTING_TABLE(SOURCE_ID,
    SCOUT_LABEL).SEND_TO;
  FORWARD_PACKET(PACKET, FORWARD_LINKS);
END GENERATE_NON_FLOOD_BROADCAST;
```

FIG. 9

```
PROCEDURE PROPAGATE_NON_FLOOD_BROADCAST(PACKET, LINK_ARRIVED_ON) IS
BEGIN
  SOURCE_ID := PACKET.SOURCE_ID;
  SCOUT_LABEL := PACKET.SCOUT_LABEL;
  IF (BROADCAST_ROUTING_TABLE(SOURCE_ID, SCOUT_LABEL).RECEIVED_FROM =
    LINK_ARRIVED_ON) THEN
    ACCEPT_PACKET(PACKET);
    FORWARD_LINKS := BROADCAST_ROUTING_TABLE(SOURCE_ID,
      SCOUT_LABEL).SEND_TO;
    FORWARD_PACKET(PACKET, FORWARD_LINKS);
  END IF;
END PROPAGATE_NON_FLOOD_BROADCAST;
```

FIG. 10

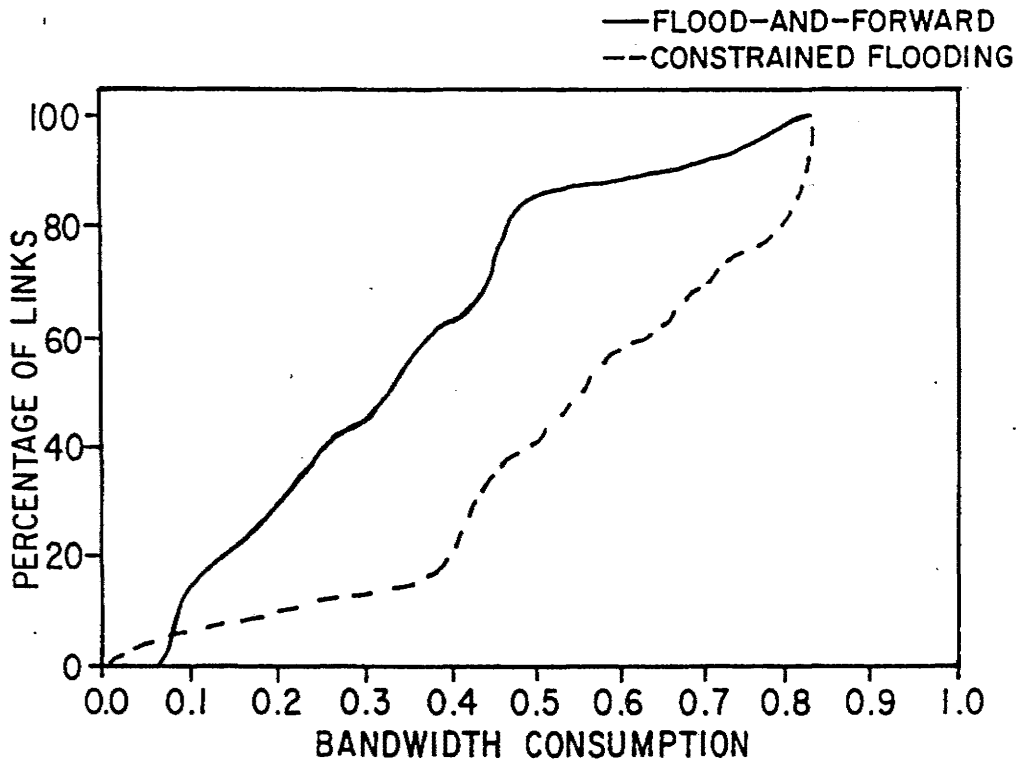


FIG. 11

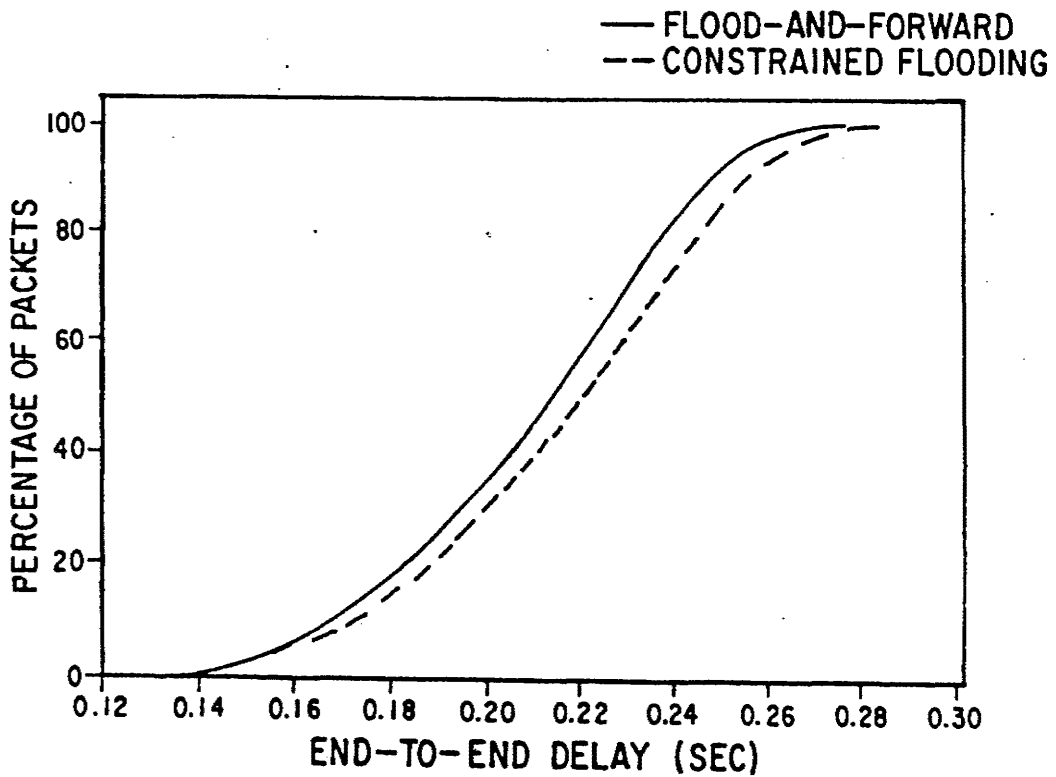


FIG. 12

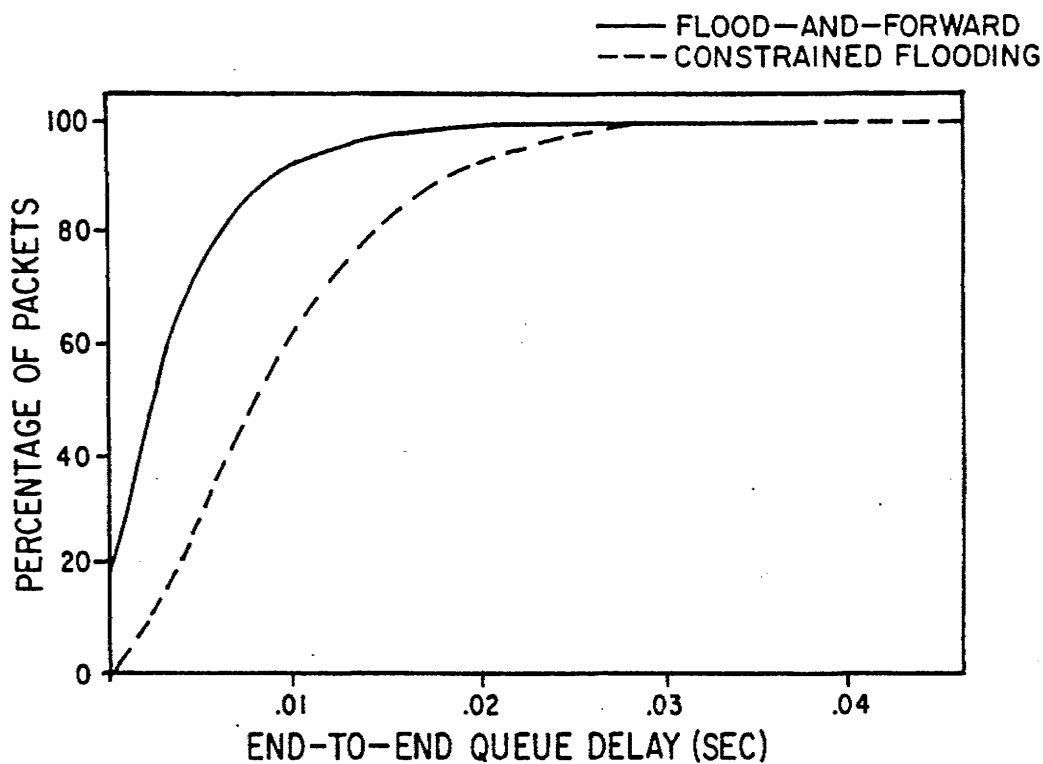


FIG. 13

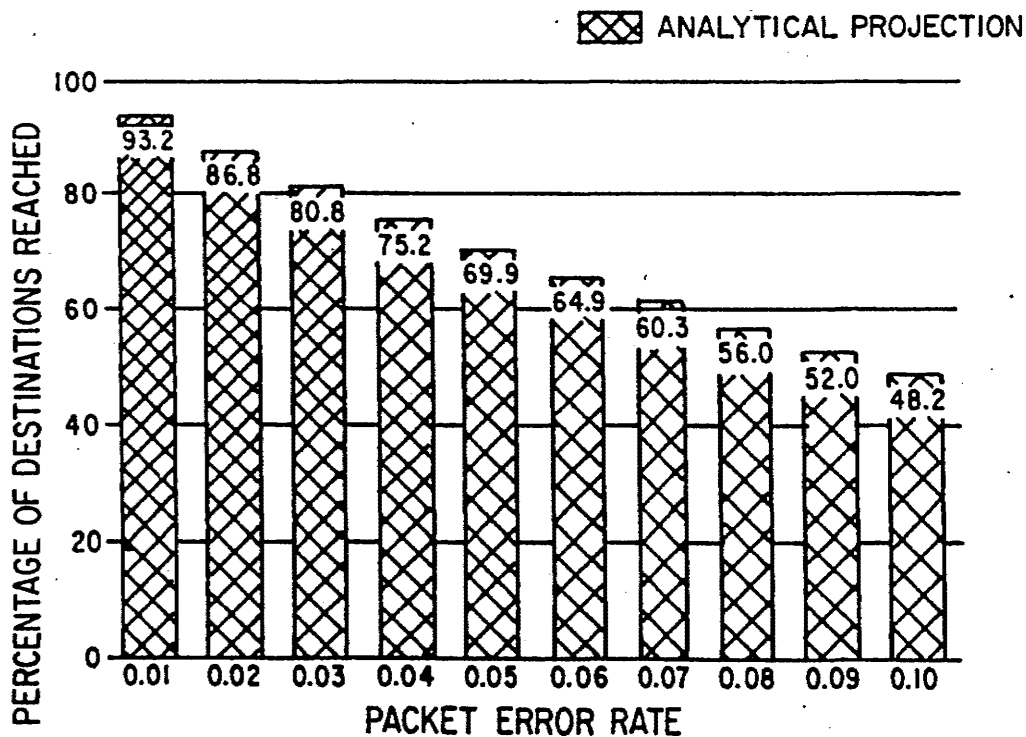


FIG. 14

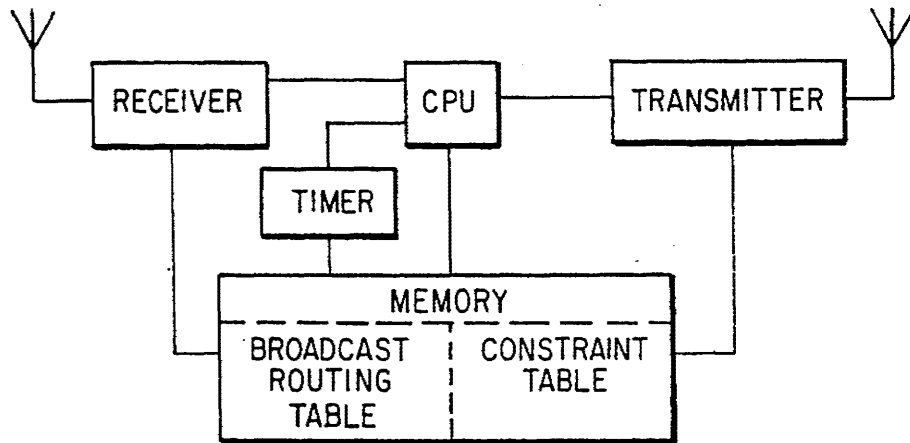


FIG. 17

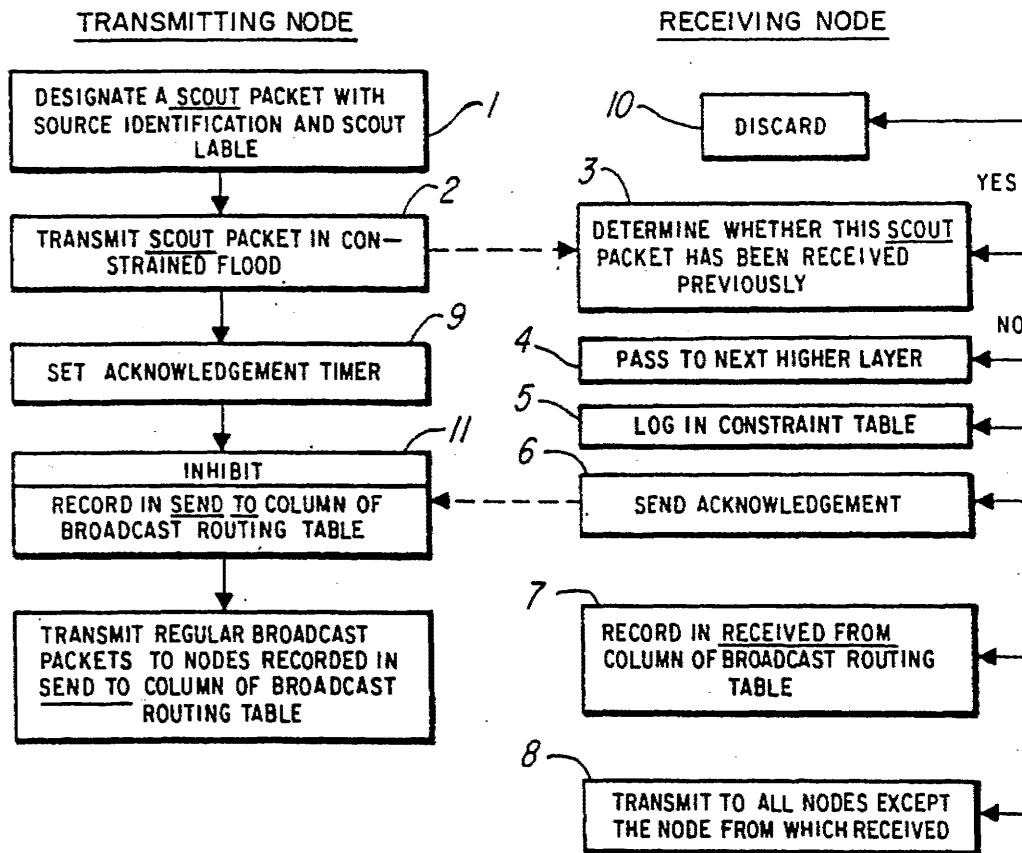


FIG. 18

FLOOD-AND-FORWARD ROUTING FOR BROADCAST PACKETS IN PACKET SWITCHING NETWORKS

This invention was made with United States government support under contract No. F30602-86-C-0224. The United States government has certain rights in this invention.

BACKGROUND OF THE INVENTION

Communication between distant stations or nodes, for example communication between nodes in a satellite network orbiting the earth, presents many requirements that have not previously been encountered in communication or computer networks. Most such communication networks are expected to be based in space and to consist of thousands of satellites orbiting the earth at various altitudes and inclinations. Such satellites are constantly moving in and out of line of sight of each other and, when engaged in a battle, are subject to all kinds of threats, ranging from link jamming to total destruction. The dynamic and volatile nature of such networks poses a great challenge to network management, requiring techniques that adapt to topological changes and that can survive threats.

In some situations, it may be desired to transmit a packet of data from one node to one specific other node. Such transmissions are referred to as "point-to-point". In other situations, it may be desired to send data from a sensor node in the network to all other nodes in the network, or to all the nodes in a subnetwork of the network. By way of illustration, in a defensive military environment, a subnetwork of sensor nodes may be in high earth orbits, as depicted in FIG. 1, while a subnetwork of weapon nodes may be in low earth orbits, as depicted in FIG. 2. It may be necessary to broadcast a packet of data from a sensor node in the high earth orbit to every weapon node in the low earth orbit. Such a transmission from one node to every node of a subnetwork rather than to a specific one or few of those nodes, is referred to as a "broadcast"; i.e., a transmission from the one node that is "broadcast" to all those other nodes. In the following description, then, "broadcast" refers to such a transmission from one node intended for all of a network or subnetwork of nodes.

In a broadcast mode, packets of data can be guided by a broadcast routing algorithm installed at each node to relay the data packets from node to node over point-to-point links so as to be received by all nodes that are reachable from the source node. This mode of broadcasting has been done before in military or commercial packet switching networks. However, in some defense scenarios, data generated by all the sensor sources can total hundreds of thousands of packets per second. Assured broadcasting of such a quantity of data requires that, in addition to being adaptive and survivable, the broadcast routing algorithm be able to handle a very high traffic load without exacting a heavy toll from the network resources.

There are many existing algorithms for routing broadcast packets in a packet switching network; however, none has been found which offers a satisfactory combination of good performance and low cost. Some algorithms excel in the selection of least delay routes at the expense of bandwidth memory, or computing power. Others sacrifice routing optimality for low overhead cost. Previous work on broadcast routing algo-

gorithms is described, for example, in *Computer Networks*, by A. S. Tanenbaum, Prentice Hall, Englewood Cliffs, N.J., 1981, and in "Reverse Path Forwarding of Broadcast Packets," by Y. K. Dalal and R. M. Metcalfe, *Communications of the ACM*, Vol. 21, pp. 1040-1048, December 1978.

One such existing algorithm is known as separate destination addressing. If the network is already equipped with a point-to-point routing algorithm, the obvious approach to broadcast routing is to have the source generate a copy of the broadcast packet for each destination and then to use point-to-point routing to deliver each copy. This approach makes good use of existing hardware; however, since routes to different destinations often overlap, most relaying nodes will receive, process and transmit the same packet over and over again. The abundance of duplicates represents a waste of bandwidth and is likely to create a high level of congestion, especially in areas close to the source.

Another approach to broadcast routing is to carry multiple destination addresses with each packet. When a broadcast packet is generated or received at a node, it partitions the remaining list of destinations, grouping together destinations that map to the same outgoing link in its point-to-point routing table. For each such group the node generates a copy of the broadcast packet, attaches the group's destination list, and forwards the packet on the selected outgoing link. As the broadcast propagates farther away from the source, the destination list gets smaller until there is no destination remaining. The packets follow the branches of a spanning tree rooted at the source, although information on the spanning tree is not explicitly stored at each node.

Forwarding broadcast packets along branches of a spanning tree has the advantage of generating an optimal number of packet copies; this number is exactly equal to the number of reachable destinations. However, the disadvantage of multidestination addressing lies in the long destination field. A bit map implementation requires as many bits as there are nodes. For networks with thousands of nodes, the high ratio of overhead bits to data bits is a drawback that is not easy to overcome. Furthermore, the performance of this algorithm is tied to that of the underlying point-to-point routing algorithm. If the point-to-point routes are inconsistent, unstable or slow to adapt, the same effects will be exhibited in the broadcast routes.

Constrained flooding is a technique in which an arriving packet that has not been seen before is copied and forwarded on all outgoing links except the link on which it arrived. Packets that have been seen are simply discarded. To keep track of already seen packets, a sequence number is assigned to each packet by the source node, and a constraint table or bit map is maintained at each node to log received packets. The log for a particular packet has to remain in the constraint table for some time, at least for the duration of the broadcast, before the log can be cleared and reused. The size of the constraint table is thus proportional to this predefined time for packets to live, the maximum traffic generation rate by all sources, and the total number of sources. For the architecture of a large network, the constraint table can take up a large chunk of memory. Estimates of millions of bits are not too exaggerated. On top of this hefty memory requirement, constrained flooding also generates a large number of packet copies— $(N)(L-1)+1$ copies, where N is the number of

nodes in the network and L is the number of links per node.

As for advantages, constrained flooding is most noted for its robustness. Packet copies that seem to be such a waste at first glance are much needed to replace lost copies. In a simulation of a 300-node network, the algorithm delivered packets to 99.83% of the destination nodes in spite of a 10% packet error rate. Constrained flooding is also known for its selection of best routes. The algorithm always finds the shortest routes possible, and its routes adapt instantaneously to changes in the network.

The least consumption of bandwidth in broadcasting is achieved with minimum spanning tree forwarding, in which packets are forwarded along branches of a spanning tree stored explicitly at each node. If the spanning tree has shortest paths from the source of the broadcast to all destinations, delay is also minimized. The biggest problem in minimum spanning tree forwarding is in determining how the nodes in the network are to generate and maintain such a tree for each source.

One accepted solution is to allow each node access to a global topology database. Every node generates and receives frequent topology updates which include not only changes in the topology but also changes in the link metric. From the information contained in its local copy of the database, each node uses the Dijkstra's shortest path first algorithm to independently compute minimum spanning trees, each one rooted at a different source. This algorithm is discussed in the above-cited article by Dalal and Metcalfe. This computation has a complexity in the order of (SN^2) operations, where S is the number of sources, and N the number of nodes in the network. For thousands of nodes and hundreds of sources, the amount of computation required in this approach is quite demanding. Moreover, keeping the distributed database consistent and up to date is not an easy task in a hostile environment. The success or failure of this task has a tremendous effect on the quality of the routes computed.

If outgoing links for any destination are taken from every point-to-point routing table in the network and joined together, they form a spanning tree from all nodes to the chosen destination. This fact is cleverly exploited in reverse path forwarding. A node forwards a received broadcast packet on all links except the incoming link if and only if the incoming link is the same link the node would use to send an addressed packet back to the source of the broadcast. Thus, this broadcast routing algorithm also makes use of a spanning tree, but this time the tree has shortest reverse paths; i.e., paths from the destinations to the source.

This approach has both advantages and disadvantages. The principal advantage is that there is no need to compute trees since tree branches are readily available in point-to-point routing tables. The major disadvantage is that the trees are inverted, implying that they may not yield shortest paths from the source to the destinations. Also, links must be flooded with packet copies in the hope that neighbors that are not part of the tree will stop the flood, although this undesirable situation can be corrected by having neighbors exchange their routing tables.

SUMMARY OF THE INVENTION

The present invention is a broadcast routing algorithm that closely matches the delay performance of constrained flooding and that provides optimal band-

width use of minimum spanning tree forwarding, yet that requires only a small amount of memory and computation. Achievement of these seemingly conflicting objectives is possible because of the following fact. As shown in three dimensions in FIG. 3a, and more clearly in a flat projection in FIG. 3b, routes taken by packets of a constrained flood constitute a minimum spanning tree from the source of the broadcast to all reachable destinations. Therefore, constrained flooding provides a fast and inexpensive means to generate a tree of shortest broadcast paths. Advantage is taken of this fact by combining with it a mechanism allowing each node to capture its individual segment of the spanning tree and to use it for forwarding of broadcast packets.

Accordingly, the flood-and-forward routing algorithm of the present invention involves periodically sending out a broadcast packet in a constrained flood broadcast. Each receiving node, the general configuration of which is depicted in FIG. 17, determines whether it has previously seen that broadcast packet. If so, it discards the packet. If not, it sends an acknowledgement of receipt of the packet back to the node that transmitted the packet to it and transmits the broadcast packet to further nodes in accordance with the constrained flood broadcast. Each node records in a broadcast routing table the other nodes from which that node receives an acknowledgement and transmits, or forwards, further broadcast packets to those same nodes until new routes are determined by the next constrained flood packet.

With its two phases, i.e. the constrained flood packet and the regular broadcast packets, this flood-and-forward routing adroitly combines constrained flooding and minimum spanning tree forwarding to provide the advantages, but not the shortcomings, of both routing algorithms.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other aspects and advantages of the present invention are more apparent from the following detailed description and claims, particularly when considered in conjunction with the accompanying drawings, in which:

FIG. 1 schematically depicts a subnetwork of sensor nodes in high earth orbits;

FIG. 2 schematically depicts a subnetwork of weapon nodes in low earth orbits;

FIG. 3a schematically depicts in three dimensions routes of a constrained flood broadcast, and FIG. 3b schematically depicts those routes in a flat projection;

FIGS. 4-10 present the program design language for various of the algorithms utilized in the present invention;

FIG. 11 graphically presents cumulative distribution function curves comparing bandwidth consumption in a transmission broadcast in accordance with the present invention and in a transmission broadcast in a constrained flood;

FIGS. 12 and 13 graphically compare transmission delays between a broadcast in accordance with the present invention and a broadcast utilizing a constrained flood;

FIGS. 14, 15, and 16 graphically depict broadcast survivability in a transmission in accordance with the present invention;

FIG. 17 is a block diagram depicting the general configuration of a node; and

FIG. 18 is a flow chart of the overall operation.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

With reference to FIG. 18, at predetermined intervals, a source node, such as one of the source nodes depicted in FIG. 1, designates a broadcast packet as a Scout packet (step 1) and transmits that packet in a constrained flood (step 2), not only to send the packet to all receiving nodes, e.g. the nodes depicted in FIG. 2, but also to establish the broadcast routes to those receiving nodes. FIG. 4 sets forth the program design language for the algorithm to determine whether it is time for a packet to be designated as a Scout packet. Each such Scout packet is identified by a Source Id, indicative of the source node, and a Scout Label, identifying the particular Scout packet from that source node.

At each receiving node, if the Scout packet has not been seen before by the node, as determined from the packet's Source Id and Scout Label (step 3), then the node accepts the Scout packet and passes it on to the next higher layer (step 4), logs the Scout packet in the node's constraint table (step 5), sends back an Ack Scout packet to the neighbor node that forwarded the Scout packet (step 6), enters identification of that neighbor node in the Received From column of the broadcast routing table indexed by Source Id and Scout Label (step 7), copies and forwards the Scout packet on all links except the link on which it arrived (step 8), and sets a timer, designated Ack Scout Timer, for receipt of acknowledgements of the Scout packet from the nodes to which the node has forwarded the Scout packet (step 9). The timer is set roughly to the longest round trip transmission time between the node and its neighbor nodes. If a reliable link service protocol is used, retransmission delay must also be included. If the receiving node has previously seen the Scout packet, as determined by finding the Source Id and Scout Label in the node's constraint table, then the node discards the Scout packet (step 10). The constraint table can be a memory look-up table. FIG. 5 sets forth the program design language for this. The Source Id and Scout Label are sufficient for proper acknowledgement of a Scout packet. Due to their brevity, Ack Scout packets may be piggybacked and redundantly transmitted for added reliability. FIG. 6 sets forth the program design language for the source node to generate the Scout packets.

Broadcast routes are extracted from Ack Scout packets and recorded as they are received by each node. When the Ack Scout Timer expires, the node enters in the Send To column of its broadcast routing table a list of its neighbor nodes from which it has received an Ack Scout packet (step 11), indexed by Source Id and Scout Label, and enters Null for those neighbor nodes from which it has not received an Ack Scout packet. FIG. 7 sets forth the program design language for this algorithm.

The same algorithm also works for the source node, which originates the flood packet, if the source node treats itself as a forwarding node and utilizes a null indication to indicate the incoming link so that the source node forwards the Scout packet on all of its links.

After the Ack Scout Timer expires at the source node, routes indexed by Source Id and Scout Label in the broadcast routing table are available for use. FIG. 8 presents the program design language for establishing these routes. Use of multiple Scout Labels allows the

source node to send out another Scout packet without having to wait for the previous Scout packet to be acknowledged. As soon as a new set of routes is available, it becomes the active set of routes at the source node, but older sets of routes are still maintained by other nodes long enough for packets already in route to reach their destinations. By reusing a number of preallocated Scout Labels, one after another, the source node manages the multiple sets of routes in an orderly fashion.

In the forwarding phase, the source node sends out broadcast packets identified by its Source Id and a Scout Label, thereby signalling to receiving nodes that these packets are to be forwarded on the routes that were built from the Scout packet with the specified Scout Label. FIG. 9 is the program design language for the source node to generate and transmit these non-flood packets. Packets are copied and forwarded by nodes according to the routing information contained in the nodes' broadcast routing tables. Thus, each node uses the Source Id and Scout Label to look up entries in its Received From and Send To columns, discards packets that did not come from neighbors on its Received From list, and forwards the other packets to those of its neighbors that are on the Send To list associated with the Source Id and Scout Label of each respective packet. FIG. 10 presents the program design language for this.

This flood-and-forward routing algorithm has been coded, tested and evaluated in a high fidelity network simulation environment developed for studying communication networks. The network has six sensor nodes in a simple ring topology, of the type depicted in FIG. 1, and 299 other nodes in a brick wall topology of 13 rings, of the type depicted in FIG. 2. Sensor data are broadcast from sensor nodes to all other nodes on point-to-point links sized at 500 Kb/s each. The six sensor nodes generate a combined traffic load of 460 Kb/s. A comparison of performance and cost between flood-and-forward routing and constrained flooding has been made.

With the exception of Scout packets, packets in flood-and-forward routing are forwarded along branches of a spanning tree. For each non-flood broadcast, only one one-hop packet is generated per destination, thus achieving the optimal use of bandwidth. In FIG. 11, cumulative distribution function (CDF) curves for bandwidth consumption are shown with all links in the network included. A point on the CDF curve shows how many links have bandwidth consumption less than or equal to the corresponding value found on the x-axis. Since the network topology under evaluation has three links per node, the majority of links have their bandwidth consumption reduced in half when flood-and-forward routing is used. In general, the reduction factor is proportional to the number of links per node minus one.

Packets in flood-and-forward routing are forwarded on routes selected by constrained flooding. However, since there are fewer packet duplicates, and thus less congestion and queue delay, end-to-end delay is better than that of constrained flooding. FIG. 12 shows CDF curves comparing the delay from broadcast source to all destinations utilizing the flood-and-forward technique of the present invention with constrained flooding. The smaller queue delay experienced by packets in flood-and-forward routing, as shown in FIG. 13, is a direct consequence of the optimal use of bandwidth.

Each source in flood-and-forward routing maintains multiple sets of routes to be able to adapt quickly to

existing traffic conditions. For example, if Scout packets are generated at a rate of 10 Scout packets per second, new routes are available every 0.1 second. The rate of generation of Scout packets can be adjusted up or down to meet network requirements.

It takes some time for a Scout packet to propagate and reach all nodes, but the source node does not have to wait for feedback from the most distant node. Routes can be used almost as soon as neighbor nodes acknowledge the Scout packet. The wait period is approximately twice the longest hop delay in the network, so as to prevent non-flood packets from catching up with Scout packets. If topological changes cause some route segment to be broken, the network will recover from any packet loss as soon as the wait period for a new Scout packet ends, e.g., within 0.1 seconds.

Broadcast survivability is a measure of how well a routing algorithm manages to deliver a broadcast packet in face of packet errors. Unfortunately, for any broadcast routing scheme employing a spanning tree, loss of a forwarded packet due to packet errors may result in a whole subtree of destinations not receiving the broadcast.

The effect of packet errors on broadcast survivability can be analytically projected for such routing schemes. For example, for a perfectly balanced binary tree, i.e., three links per node, let p be the packet error rate, and $2^N - 1$ the number of nodes. Then, the average number of nodes not receiving a broadcast is given by:

$$\sum_{i=0}^{N-1} (1-p)^i (2^N - 2^i)$$

The percentage of nodes receiving a broadcast is derived from this equation and plotted as a function of the packet error rate in FIG. 14. The projection appears to show disappointing broadcast survivability for flood-and-forward routing; however, when flood-and-forward routing is simulated with a reliable link service protocol, which implements hop-by-hop acknowledgment and retransmission, the broadcast quality is vastly improved to about the same level as that of constrained flooding, as depicted in FIG. 15. For applications which depend critically on the broadcast being received by every node, flood-and-forward routing guarantees a very high probability of perfect delivery. For a 10% packet error rate, FIG. 16 shows 98.52% perfect delivery by flood-and-forward routing, as contrasted with 66.58% perfect delivery by constrained flooding.

The true strength of flood-and-forward routing is that it achieves its superb performance with very little cost. A Scout packet is just a data packet spearheading a wave of other data packets, and routes are generated as needed by each source. A Scout packet will not be sent if there is no traffic, but if the routes are outdated, the first data packet out must be a Scout packet. The overhead in Ack Scout packets is insignificant, and there is no computation and maintenance of spanning trees.

In flood-and-forward routing, the size of the constraint table is proportional to the Scout packet generation rate (which is comparatively very low), time for Scout packets to live, and 0 the number of sources. For a traffic rate of 50,000 pkts/sec and a Scout packet rate of 10 packets/sec., this represents a size reduction of 5000 to 1. Similarly, the size of the broadcast routing table is proportional to the Scout packet generation

rate, the time for routes to live, the number of sources, and the number of links per node.

What is claimed is:

1. In a communication network having a plurality of communication nodes including at least one source node and a plurality of receiving nodes, the source node transmitting data packets to all the receiving nodes in a broadcast transmission mode, a method of operation comprising:
 - (a) at the source node, periodically designating one of the data packets as a scout packet for establishing broadcast routes to the receiving nodes; incorporating into the scout packet a source identification, indicative of the identification of the source node, and a scout label, indicative of the particular scout packet; transmitting that scout packet to all the receiving nodes in a constrained flood broadcast transmission; and initiating a first time interval having a predetermined duration for receipt of acknowledgements of receipt of the scout packet;
 - (b) at each receiving node, maintaining a constraint table of the source identifications and scout labels of received scout packets and a broadcast routing table for received scout packets, said broadcast routing table being indexed by source identifications and scout labels; receiving the transmitted scout packet; determining whether the source identification and scout label of the received scout packet are in the constraint table; if the source identification and scout label of the received scout packet are in the constraint table, then discarding the scout packet; and if the source identification and scout label of the received scout packet are not in the constraint table, then (1) transmitting an acknowledgement of the scout packet to the node from which the scout packet was received, (2) transmitting the scout packet to other receiving nodes in accordance with the constrained flood broadcast transmission, (3) initiating a second time interval having a predetermined duration for receipt of acknowledgements of receipt of the scout packet by said other receiving nodes, (4) recording the source identification and scout label of the scout packet in the constraint table, and (5) recording in a "received column" in the broadcast routing table the identification of the node from which the scout packet was received, and (6) recording in a "send to" column in the broadcast routing table the identification of the other receiving nodes from which an acknowledgement of receipt of the scout packet is received during the second time interval;
 - (c) at the source node, receiving acknowledgements of receipt of transmitted scout packets; incorporating into non-scout data packets the source identification and scout label of the scout packet for which the first time interval has most recently expired; and transmitting the non-scout data packets to those nodes from which an acknowledgement has been received of receipt of the scout packet having the source identification and scout label that are incorporated into such non-scout data packet; and
 - (d) at each receiving node, receiving non-scout data packets; determining whether the "received from" column of the broadcast routing table has recorded therein as the node from which said receiving node received the scout packet having the same source identification and scout label as are incorporated in the received non-scout data packet the identifica-

9

tion of the node from which said each receiving
 node received the non-scout-data packet; if the
 "received from" column of the broadcast routing
 table has recorded as the node from which said
 receiving node received the scout packet having
 the same source identification and scout label as are
 incorporated in the received non-scout data packet
 the node from which the receiving node received
 the non-scout data packet, then transmitting the
 non-scout data packet to those receiving nodes
 recorded in the "send to" column of the broadcast
 routing table for that scout packet; and if the "re-
 ceived from" column of the broadcast routing table
 does not have recorded as the node from which
 said receiving node received the scout packet hav-
 ing the same source identification and scout label as

10

are incorporated in the received non-scout data
 packet the node from which the receiving node
 received the non-scout data packet, then discarding
 the non-scout data packet.

2. A method as claimed in claim 1, wherein each
 receiving node initiates the second time interval to have
 a time substantially equal to the longest round trip trans-
 mission time between the receiving node and the other
 receiving nodes to which said receiving node transmits
 the scout packet.

3. A method as claimed in claim 1, wherein each
 receiving node transmits the source identification and
 scout label of the received scout packet as the acknowl-
 edgement.

* * * * *

20

25

30

35

40

45

50

55

60

65



UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

ASSISTANT SECRETARY AND COMMISSIONER
OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

#17

CHANGE OF ADDRESS/POWER OF ATTORNEY

FILE LOCATION 21C1 SERIAL NUMBER 09629576 PATENT NUMBER

THE CORRESPONDENCE ADDRESS HAS BEEN CHANGED TO CUSTOMER # 25096

THE PRACTITIONERS OF RECORD HAVE BEEN CHANGED TO CUSTOMER # 25096

THE FEE ADDRESS HAS BEEN CHANGED TO CUSTOMER # 25096

ON 02/21/03 THE ADDRESS OF RECORD FOR CUSTOMER NUMBER 25096 IS:

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE WA 98111-1247

AND THE PRACTITIONERS OF RECORD FOR CUSTOMER NUMBER 25096 ARE:

28006	28516	31646	33273	33904	34047	34807	36878	37263	37337
37376	37953	38264	38563	39145	42216	43449	43834	43985	44785
44865	44934	45769	46962	47392	47724	47994	48390	50213	50774
51086	51638	51770	52396						

RECEIVED
MAR 03 2003
Technology Center 2100

PTO INSTRUCTIONS: PLEASE TAKE THE FOLLOWING ACTION WHEN THE CORRESPONDENCE ADDRESS HAS BEEN CHANGED TO CUSTOMER NUMBER: RECORD, ON THE NEXT AVAILABLE CONTENTS LINE OF THE FILE JACKET, 'ADDRESS CHANGE TO CUSTOMER NUMBER'. LINE THROUGH THE OLD ADDRESS ON THE FILE JACKET LABEL AND ENTER ONLY THE 'CUSTOMER NUMBER' AS THE NEW ADDRESS. FILE THIS LETTER IN THE FILE JACKET. WHEN ABOVE CHANGES ARE ONLY TO FEE ADDRESS AND/OR PRACTITIONERS OF RECORD, FILE LETTER IN THE FILE JACKET. THIS FILE IS ASSIGNED TO GAU 2155.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/629,576	07/31/2000	Fred B. Holt	030048001	5408

25096 7590 02/04/2004

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

EXAMINER

WON, YOUNG N

ART UNIT	PAPER NUMBER
2155	8

2155

8

DATE MAILED: 02/04/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

OK

DETAILED ACTION

1. Claims 1-49 have been examined and are pending with this action.

Specification

2. This application does not contain a brief summary of the invention as required by 37 CFR 1.73. A brief summary of the invention is required.
3. This application does not contain an abstract of the disclosure as required by 37 CFR 1.72(b). An abstract on a separate sheet is required.
4. The following guidelines illustrate the preferred layout for the specification of a utility application. These guidelines are suggested for the applicant's use.

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

- (a) TITLE OF THE INVENTION.
- (b) CROSS-REFERENCE TO RELATED APPLICATIONS.
- (c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.
- (d) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC (See 37 CFR 1.52(e)(5) and MPEP 608.05. Computer program listings (37 CFR 1.96(c)), "Sequence Listings" (37 CFR 1.821(c)), and tables having more than 50 pages of text are permitted to be submitted on compact discs.) or

Art Unit: 2155

REFERENCE TO A "MICROFICHE APPENDIX" (See MPEP § 608.05(a). "Microfiche Appendices" were accepted by the Office until March 1, 2001.)

(e) BACKGROUND OF THE INVENTION.

(1) Field of the Invention.

(2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.

(f) BRIEF SUMMARY OF THE INVENTION.

(g) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).

(h) DETAILED DESCRIPTION OF THE INVENTION.

(i) CLAIM OR CLAIMS (commencing on a separate sheet).

(j) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).

(k) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) do not apply to the examination of this application as the application being examined was not (1) filed on or after November 29, 2000, or (2) voluntarily published under 35 U.S.C. 122(b). Therefore, this application is examined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

Art Unit: 2155

5. Claims 1-49 are rejected under 35 U.S.C. 102(e) as being anticipated by McCanne (US 6611872 B1).

INDEPENDENT:

As per claim 1, McCanne teaches a computer network having a plurality of participants (see title and Fig.1), each participant having connections to at least three neighbor participants (see Fig.1), wherein an originating participant sends data to the other participants by sending the data through each of its connections to its neighbor participants (see col.6, lines 27-36) and wherein each participant sends data that it receives from a neighbor participant to its other neighbor participants (see col.12, lines 60-67).

As per claim 13, McCanne teaches a component for controlling communications of a participant with a broadcast channel (see title and abstract), comprising: a contact module that locates a portal computer and requests the located portal computer to provide an indication of neighbor participants to which the participant can be connected (see col.3, lines 21-27 and col.8, lines 53-61); and a join module that receives the indication of neighbor participants and establishes a connection between the participant and each of the indicated neighbor participants (see col.6, line 65 to col.7, line 1; col.8, lines 57-61; and col.10, lines 45-48).

As per claim 19, McCanne teaches a broadcast channel for participants (see Fig.1), comprising: a communications network that provides peer-to-peer communications between the participants connected to the broadcast channel (see col.5, lines 32-35; col.7, lines 22-25; and col.23, lines 22-25); and for each participant connected to the broadcast channel, an indication of four neighbor participants of that participant (see Fig.1); and a broadcast component that receives data from a neighbor participant using the communications network and that sends the received data to its other neighbor participants to effect the broadcasting of the data to each participant of the to broadcast channel (see col.7, lines 20-22 and col.12, lines 60-67).

Art Unit: 2155

As per claim 28, McCanne teaches a broadcast channel comprising a plurality of participants (see Fig.1), each participant being connected to neighbor participants (see Fig.1), the participants and connections between them forming an m -regular graph (see Fig.1), where m is greater than 2 and the number of participants is greater than m (see Fig.1).

As per claim 36, McCanne teaches a broadcast channel comprising a plurality of participants (see Fig.1), each participant being connected to neighbor participants (see Fig.1), the participants and connections between them form an m -regular graph (see Fig.1), where m is greater than 2 (see Fig.1), and wherein when a participant receives data from a neighbor participant, it sends the data to its other neighbor participants (see col.7, lines 20-22 and col.12, lines 60-67).

As per claim 44, McCanne teaches a computer-readable medium containing instructions for controlling communications of a participant of a broadcast channel (see col.2, lines 42-44 and col.22, lines 28-37), by a method comprising: locating a portal computer; requesting the located portal computer to provide an indication of neighbor participants to which the participant can be connected; receiving the indications of the neighbor participants; and establishing a connection between the participant and each of the indicated neighbor participants (see claim 13 rejection above).

DEPENDENT:

As per claim 2, McCanne further teaches wherein each participant is connected 2 to 4 other participants (see Fig.1).

As per claims 3, 30, and 39, McCanne further teaches wherein each participant is connected to an even number of other participants (see Fig.1 and col.23, lines 25-31).

As per claims 31 and 40, McCanne further teaches wherein m is odd and the number of participants is even (see Figs.1, 2, & 3c).

As per claim 37, McCanne further teaches wherein the number of participants is greater than m (see claim 28 rejection above).

As per claims 4-6, 29, and 38, McCanne further teaches wherein the network or graph is m -regular and m -connected, where m is the number of neighbor participants of each participant (see Fig.1).

As per claims 7 and 27, McCanne further teaches wherein all the participants are peers (see col.7, lines 22-25 and col.12, lines 39-41).

As per claim 8, McCanne further teaches wherein the connections are peer-to-peer connections (see claim 19 rejection above).

As per claims 9, 18, 25, 34, 43, and 49, McCanne further teaches wherein the connections are TCP/IP connections (see col.27, line 16 and col.28, lines 28-36).

As per claims 10, 14, 15, 22, 32, 41, 45, and 46, McCanne further teaches wherein each participant is a process executing on a different computer system (see col.2, lines 42-44).

As per claim 11, McCanne further teaches wherein a computer hosts more than one participant (see col.22, lines 35-46).

As per claim 12, McCanne further teaches wherein each participant, sends to each of its neighbors, only one copy of the data (see col.19, lines 44-61).

As per claims 16, 35, and 47, McCanne further teaches of a broadcast module that receives data from a neighbor participant of the participant and transmits the received data to the other neighbor participants (see col.7, lines 20-22 and col.12, lines 60-67).

As per claims 17 and 48, McCanne further teaches of a connection request module that receives a request to connect to another participant, disconnects from a neighbor participant, and connects to the other participant (see col.19, lines 1-9).

Art Unit: 2155

As per claim 20, McCanne further teaches wherein the broadcast component disregards received data that it has already sent to its neighbor participants (see col.19, lines 44-61).

As per claim 21, McCanne further teaches wherein a participant connects to the broadcast channel by contacting a participant already connected to the broadcast channel (see col.6, line 65 to col.7, line 1; col.8, lines 57-61; and col.10, lines 45-48).

As per claim 23, McCanne further teaches wherein each participant is a computer thread (see col.2, lines 42-44).

As per claims 24, 33, and 42, McCanne further teaches wherein each participant is a computer (see col.3, lines 18-20).

As per claim 26, McCanne further teaches wherein the communications network is the Internet (see col.2, lines 45-49).

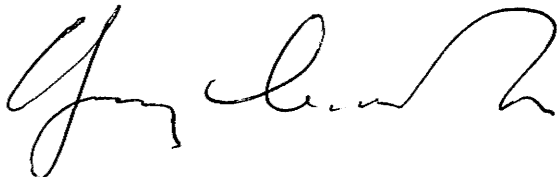
Art Unit: 2155

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Young N Won whose telephone number is 703-605-4241. The examiner can normally be reached on M-Th: 8AM-6PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hosain T Alam can be reached on 703-308-6662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Young N Won



January 29, 2004


HOSAIN ALAM
SUPERVISORY PATENT EXAMINER

84

Office Action Summary

Application No.

09/629,576

Applicant(s)

HOLT ET AL.

Examiner

Young N Won

Art Unit

2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 31 July 2000.
- 2a) This action is FINAL.
- 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-49 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-49 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) All b) Some * c) None of:
1. Certified copies of the priority documents have been received.
2. Certified copies of the priority documents have been received in Application No. _____.
3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
* See the attached detailed Office action for a list of the certified copies not received.
- 13) Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.
a) The translation of the foreign language provisional application has been received.
- 14) Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

- 1) Notice of References Cited (PTO-892)
- 2) Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) Information Disclosure Statement(s) (PTO-1449) Paper No(s) 5 & 6.
- 4) Interview Summary (PTO-413) Paper No(s). _____
- 5) Notice of Informal Patent Application (PTO-152)
- 6) Other:

Notice of References Cited

Application/Control No. 09/629,576	Applicant(s)/Patent Under Reexamination HOLT ET AL.	
Examiner Young N Won	Art Unit 2155	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-5,953,318 A	09-1999	Nattkemper et al.	370/236
	B	US-6,611,872 B1	08-2003	McCanne, Steven	709/238
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

T 1



US005953318A

United States Patent [19]

[11] Patent Number: 5,953,318

Nattkemper et al.

[45] Date of Patent: Sep. 14, 1999

[54] DISTRIBUTED TELECOMMUNICATIONS SWITCHING SYSTEM AND METHOD

[75] Inventors: Dieter H. Nattkemper, Rohnert Park; Farzad S. Nabavi, Petaluma, both of Calif.

[73] Assignee: Alcatel USA Sourcing, L.P., Plano, Tex.

[21] Appl. No.: 08/984,791

[22] Filed: Dec. 4, 1997

Related U.S. Application Data

[60] Provisional application No. 60/032,609, Dec. 4, 1996.

[51] Int. Cl.⁶ H04L 12/56

[52] U.S. Cl. 370/236; 370/396

[58] Field of Search 370/230, 231, 370/235, 236, 255, 257, 258, 395, 396, 397, 402, 409, 420

[56] References Cited

U.S. PATENT DOCUMENTS

5,301,273	4/1994	Konishi	395/200
5,528,591	6/1996	Lauer	370/60.1
5,649,110	7/1997	Ben-Nun et al.	395/200.19
5,734,825	3/1998	Lauck et al.	395/200.13

OTHER PUBLICATIONS

Rubin, I et al., "Input Rate Flow Control for High Speed Communications Networks Using Burst Level Feedback Control", *European Transactions on Telecommunications and Related Technologies*, vol. 5, No. 1, Jan. 1, 1994, pp. 107-123, odd pages only.

Kung, H.T., et al., "Credit-Based Flow Control for ATM Networks", *IEEE Network: The Magazine of Computer Communications*, vol. 9, No. 2, Mar. 1, 1995, pp. 40-48, even pages only.

Albertengo, G., "An Optimal Paket Switch Architecture for ATM 1", *Networking in the Nineties*, Bal Harbour, Apr. 7-11, 1991, vol. 2, No. CONF. 10, Apr. 7, 1991, pp. 441, 442 & 444.

Nen-Fu Huang, et al., "A Cell-Based Flow Control Scheme for DQDB Interconnections Across N ATM Switch", *Serving Humanity Through Communications*, Supercomm/ICC, New Orleans, May 1-5, 1994, vol. 1, May 1, 1994, pp. 51-56, odd pages only.

Kennington, "Credit Priority Queuing for ATM Switches", *Australian Communication Networks and Applications Conference*, 1994, Dec. 5, 1994, pp. 399-403, odd pages only.

Floyd Backes, "Spanning Tree Bridges. Transparent Bridges for Interconnection of IEEE 802 LANs", *IEEE Network*, vol. 2, No. 1, Jan. 1988, pp. 5-9.

Paul F. Tsuchiya, "The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks", *Computer Communications Review*, vol. 18, No. 4, Aug. 1988, pp. 35-42.

Adrian Segall, et al., "Reliable Multi-User Tree Setup with Local Identifiers", *IEEE Infocom '92*, vol. 3, Conf. 11, Jan. 1, 1992, pp. 2096-2106.

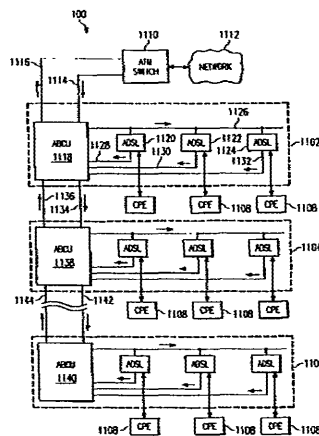
(List continued on next page.)

Primary Examiner—Melvin Marcelo
Attorney, Agent, or Firm—Fliesler, Dubb, Meyer & Lovejoy

[57] ABSTRACT

A distributed telecommunications switching system (100) is disclosed. The system includes a controller (118) in a first switching system (102). The control signal is transmitted to one or more intermediate switching system (104) and a terminating switching system (106) that generates and transmits a control signal which carries a plurality of credit allowance values. The intermediate switching system (104) receives the control signal and transmits data packets to the first switching system (102) in response to a first credit allowance value. The terminating switching system (106) receives the control signal and transmits data packets to the first switching system (102) in response to a second credit allowance value. The first switching system (102) transmits data packets from the intermediate switching system (104) and the terminating switching system (106) along with its own data packets according to its credit value to a data packet switch (110).

29 Claims, 11 Drawing Sheets



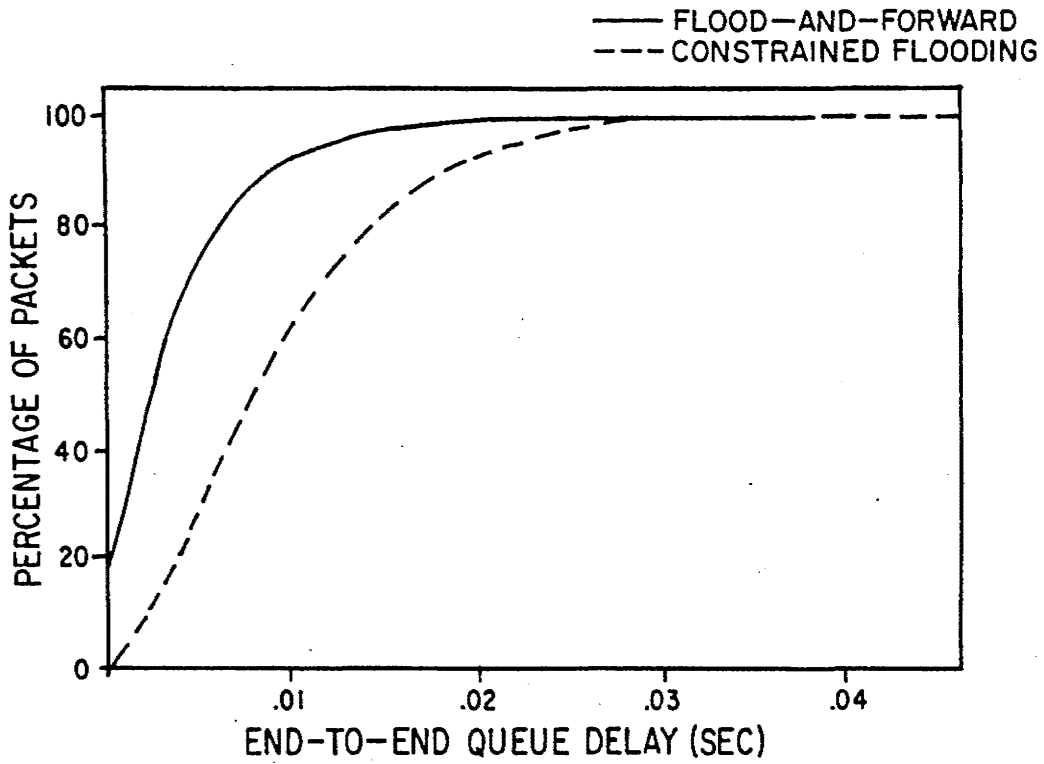


FIG. 13

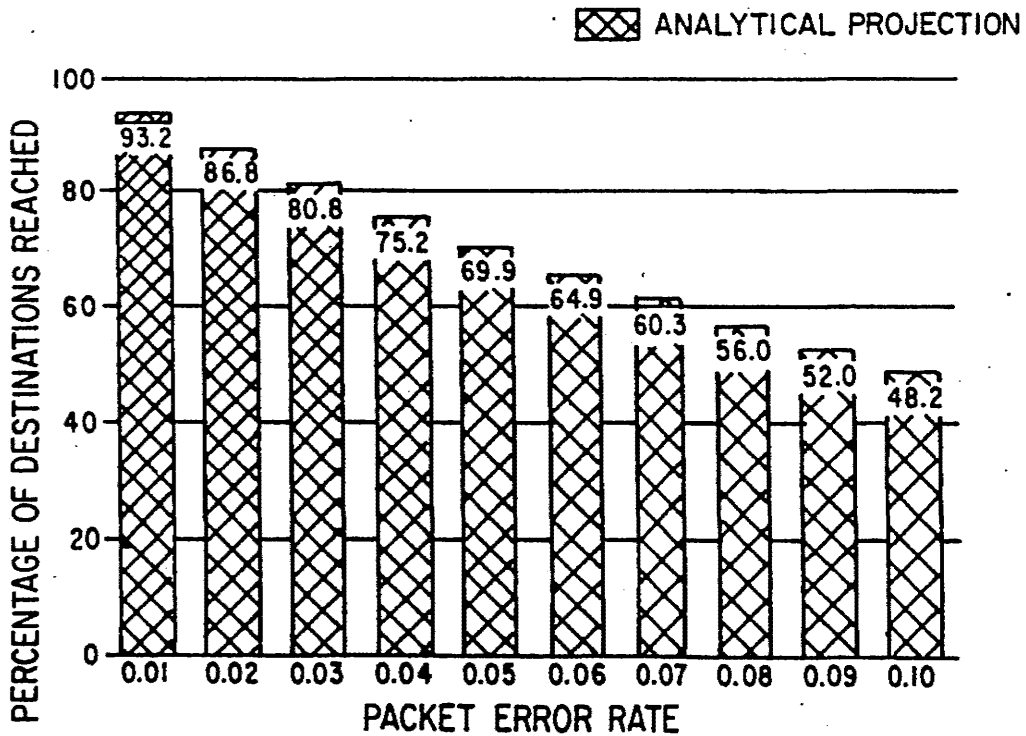


FIG. 14

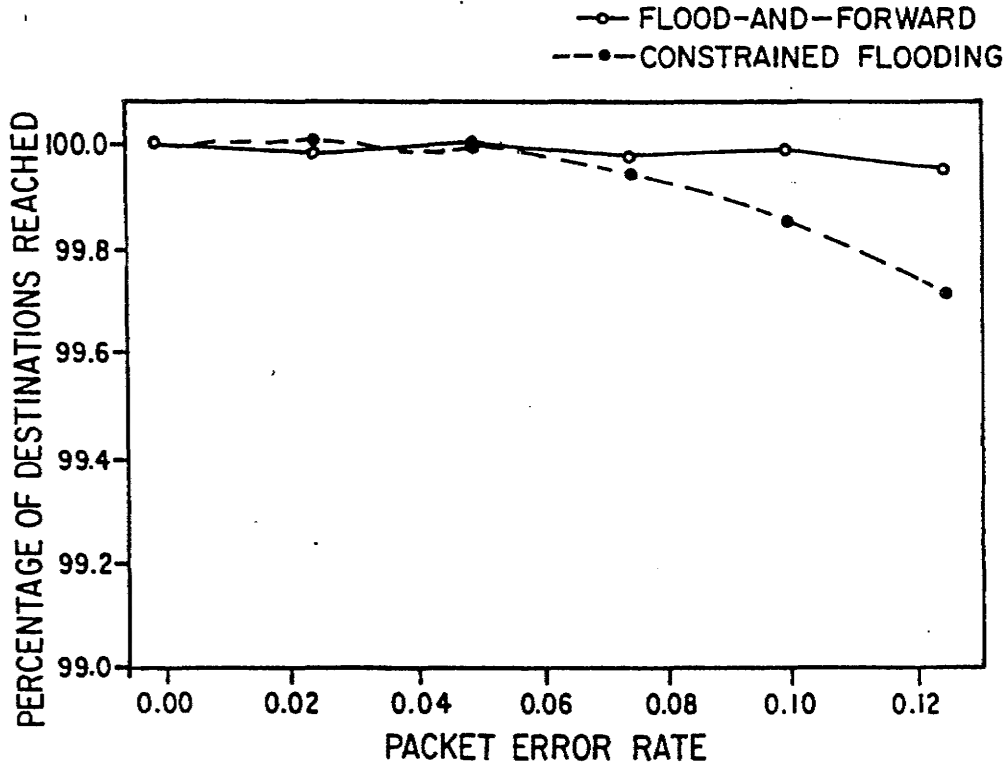


FIG.15

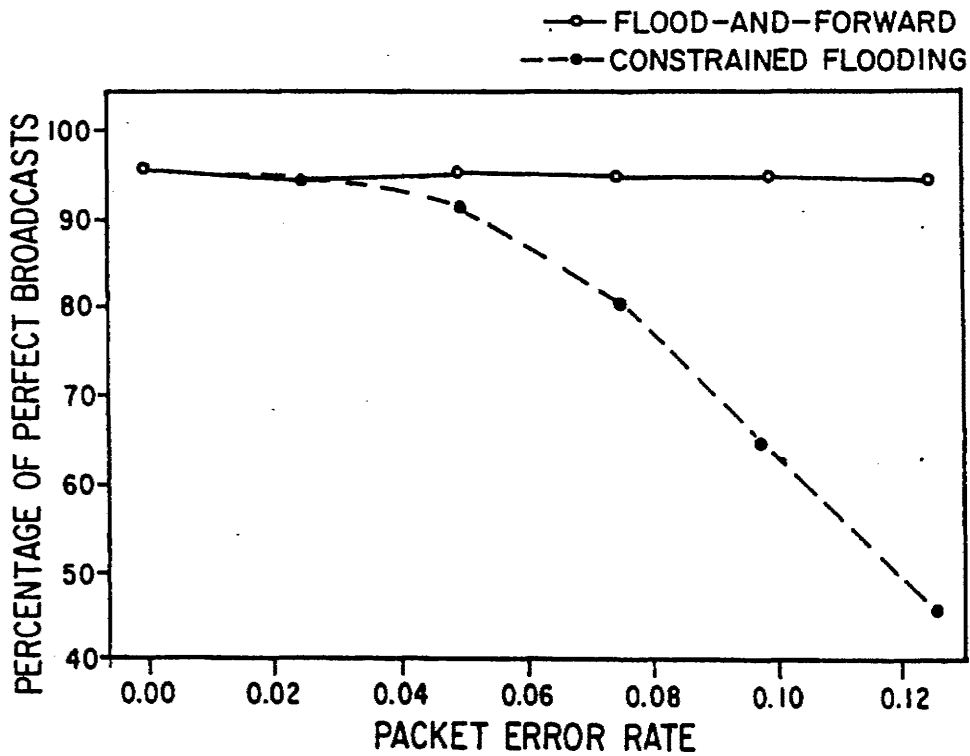


FIG.16

OTHER PUBLICATIONS

Michael D. Schroeder, et al., "A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," *IEEE Journal on Selected Areas in Communications*, vol. 9, No. 8, Oct. 1, 1991, pp. 1318-1335.

Keiichi Koyanagh, et al., "Distributed Communications System Technology," *IEICE Transactions on Communications*, vol. E77-B, No. 11, Nov. 1, 1994, pp. 1350-1362.

Kang G. Shin, et al., "A Simple Distributed Loop-Free Routing Strategy for Computer Communication Networks," *IEEE Transactions on Parallel & Distributed Systems*, vol. 4, No. 12, Dec. 1, 1993, pp. 1308-1319.

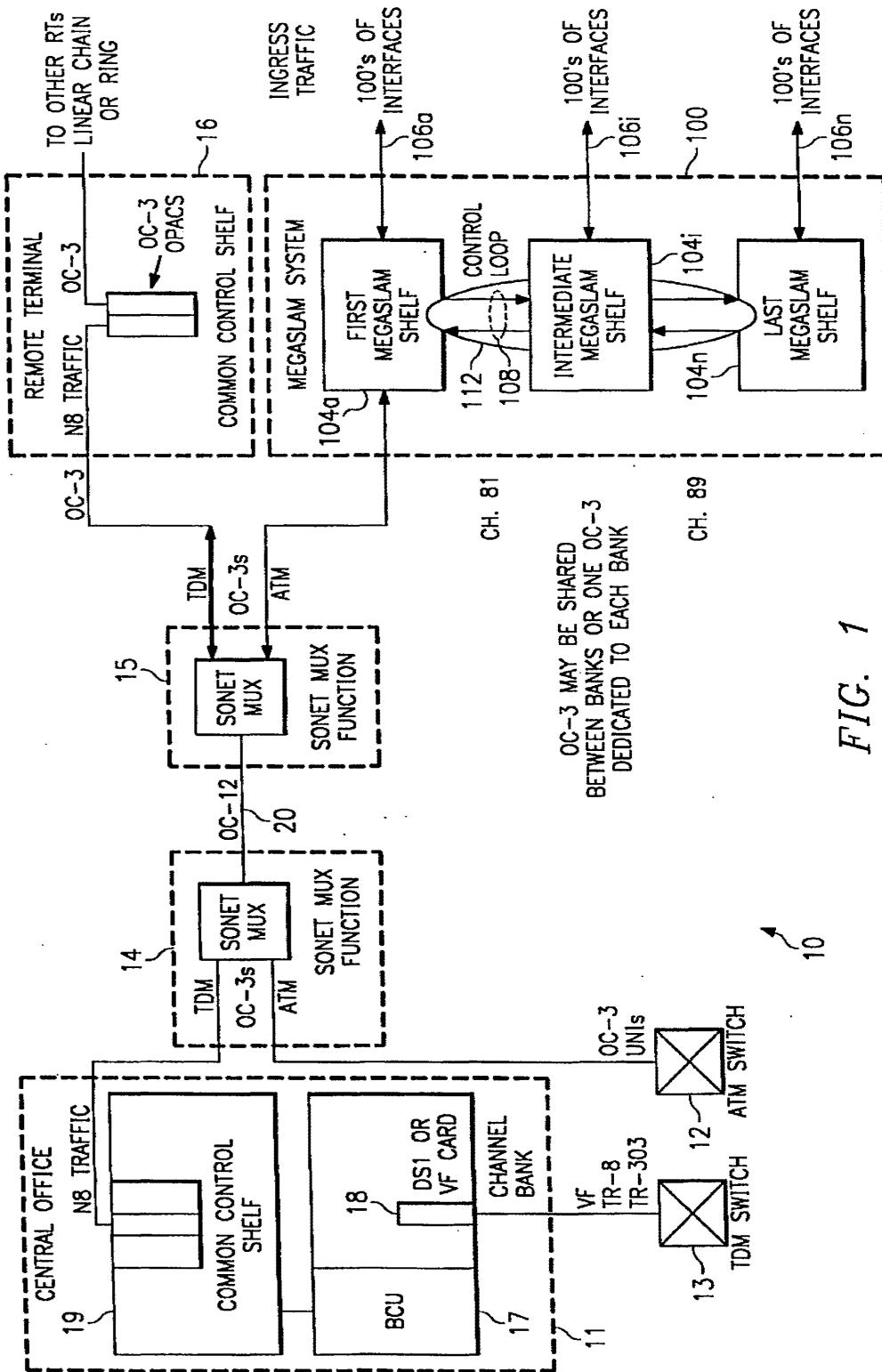
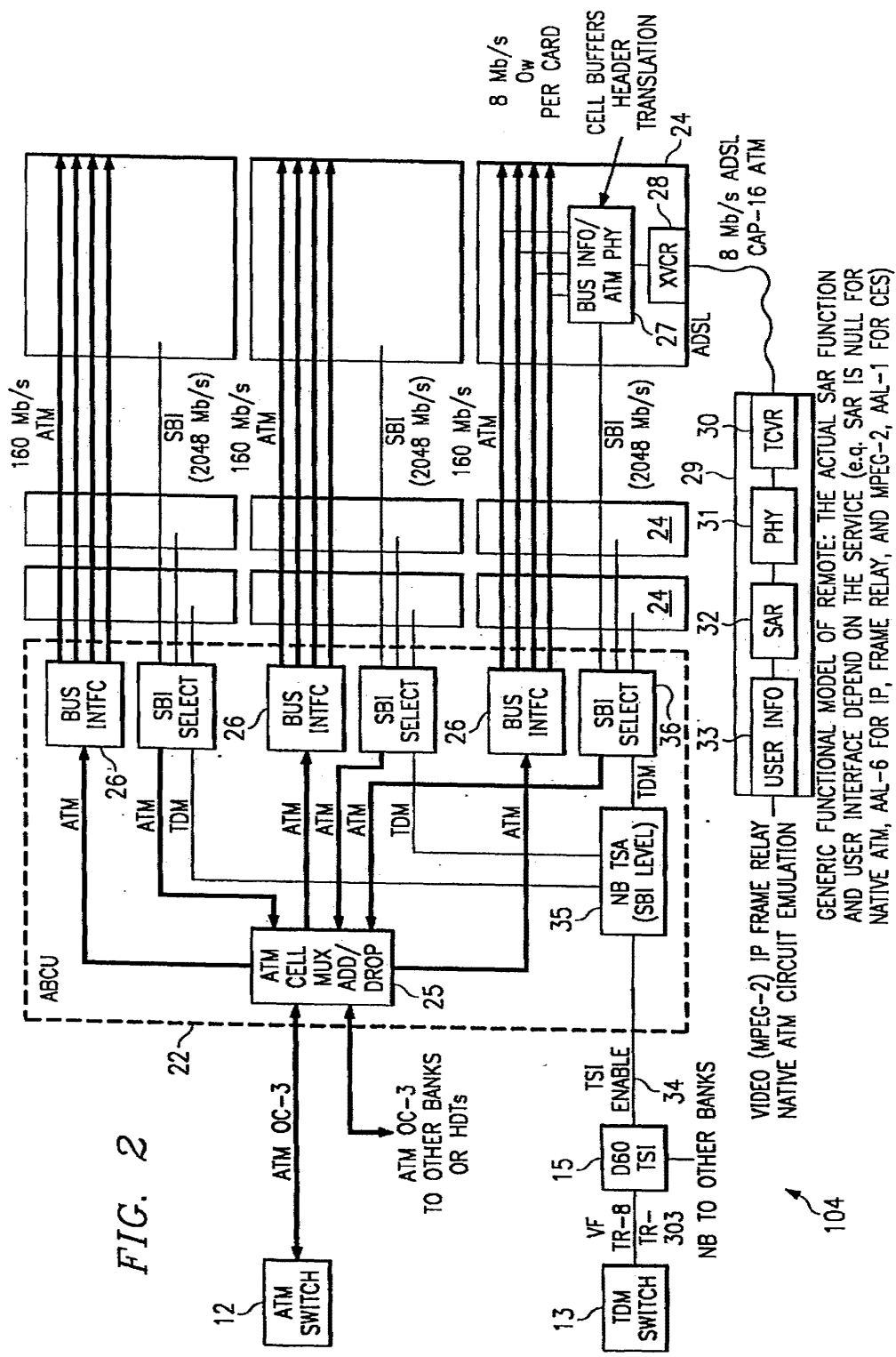


FIG. 1



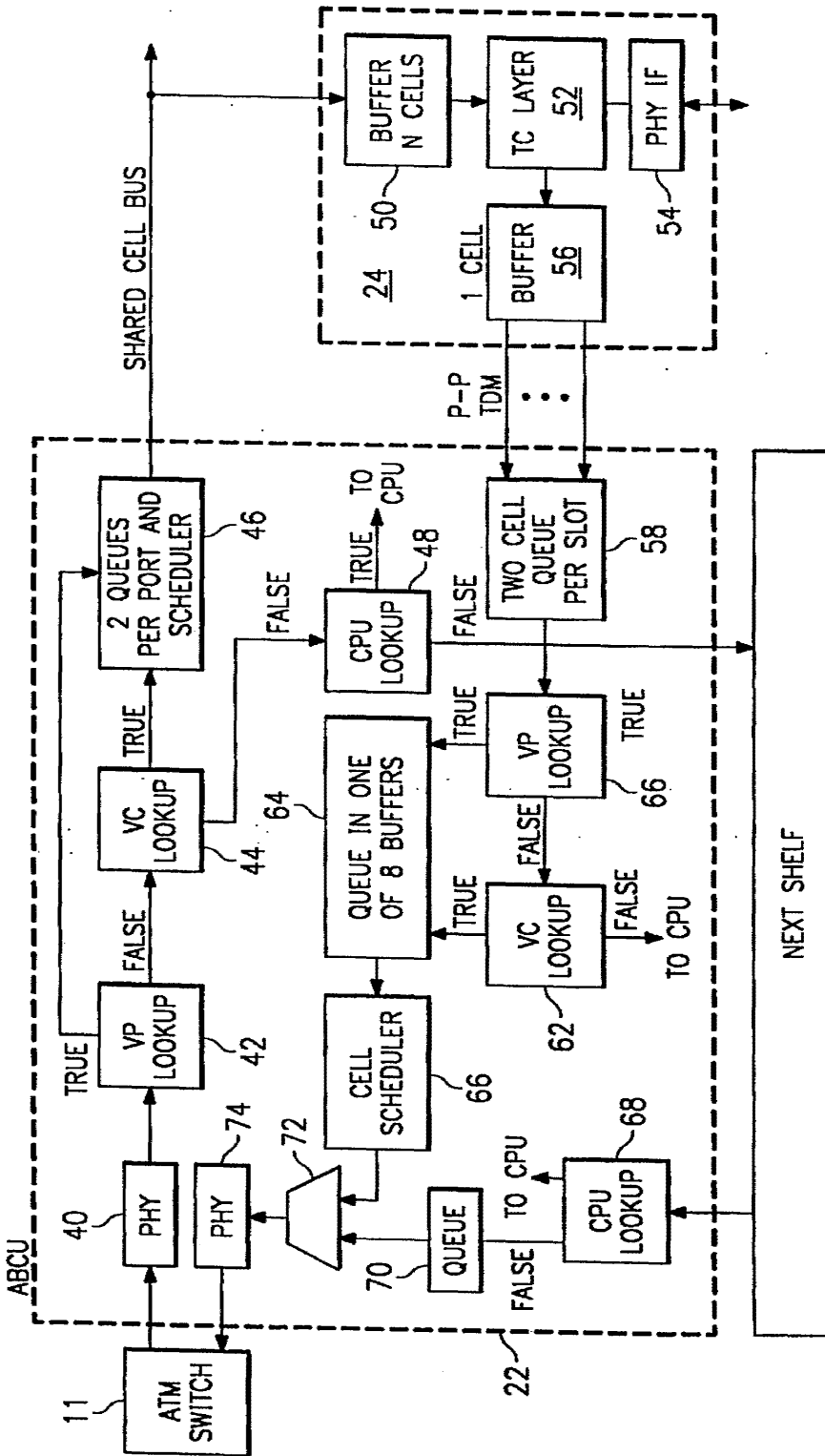


FIG. 3

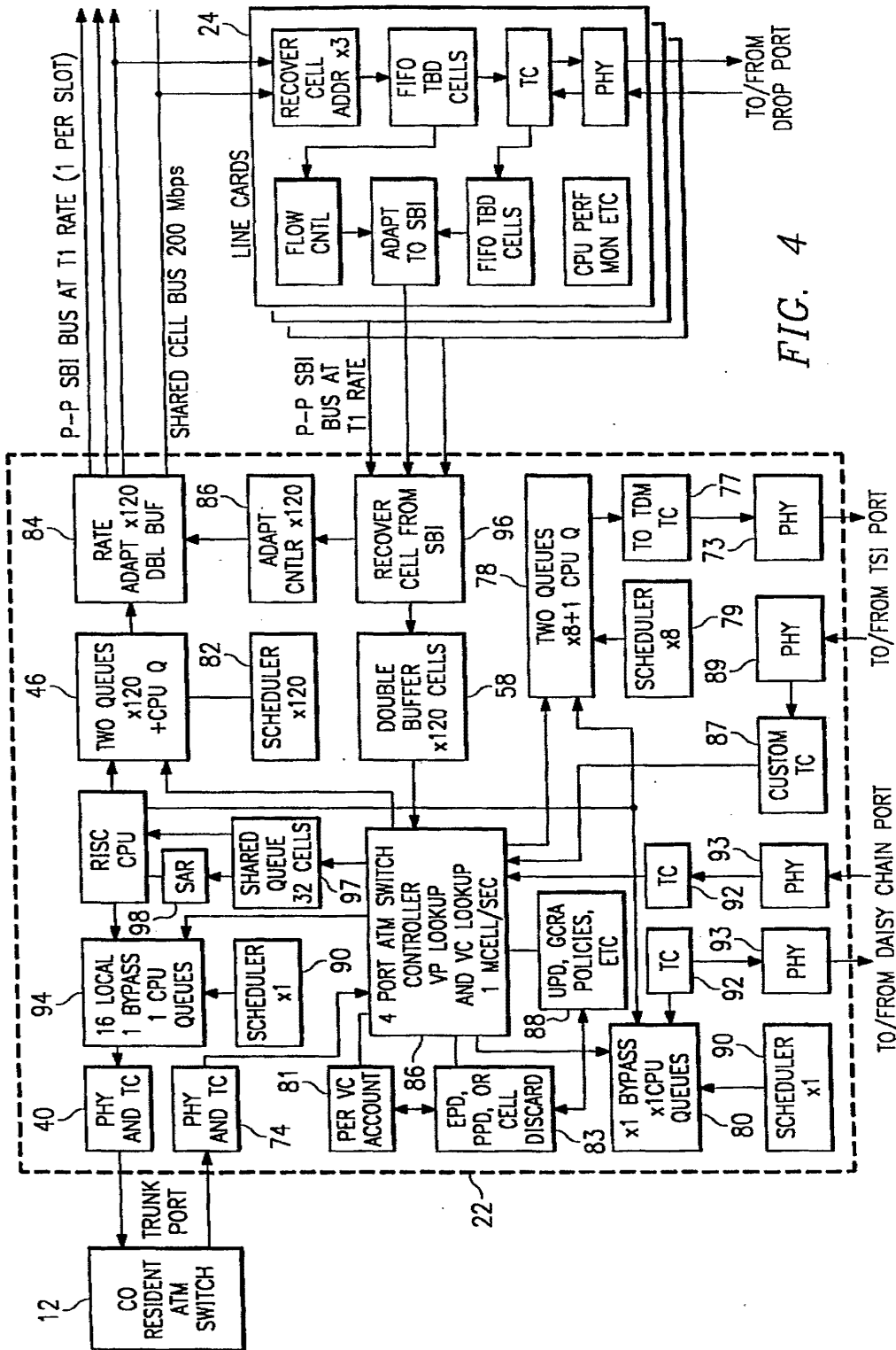


FIG. 4

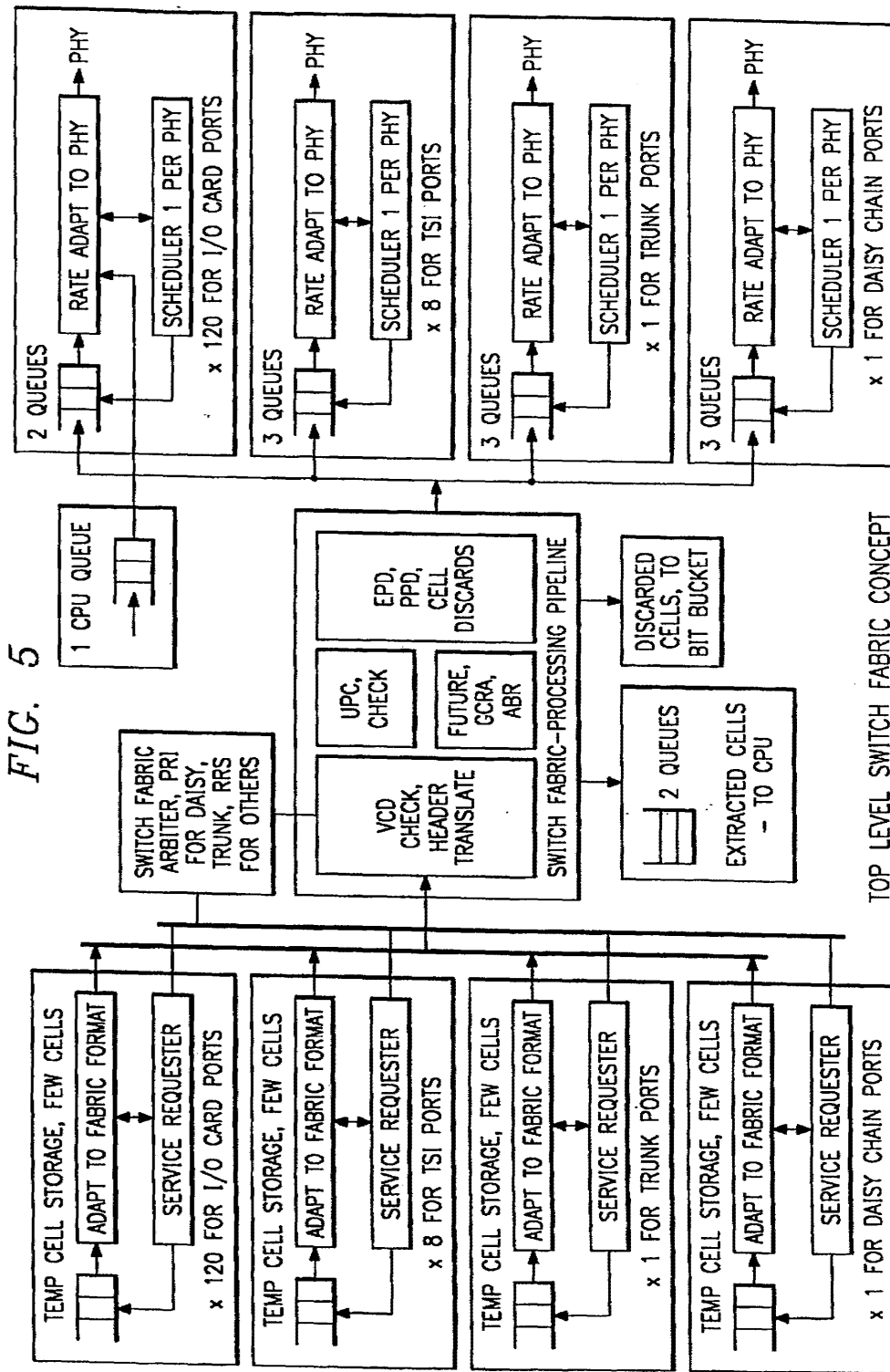
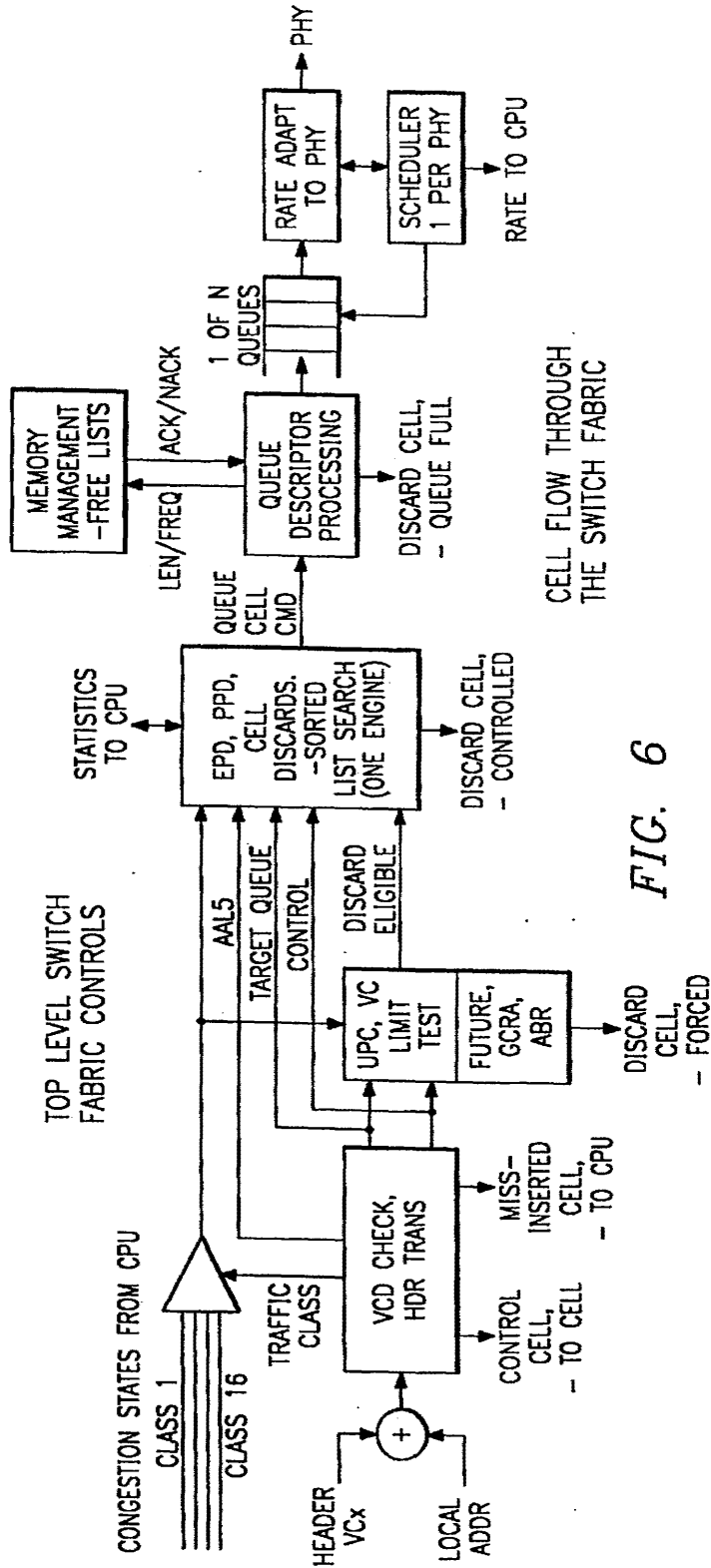


FIG. 5

TOP LEVEL SWITCH FABRIC CONCEPT



CELL FLOW THROUGH THE SWITCH FABRIC

FIG. 6

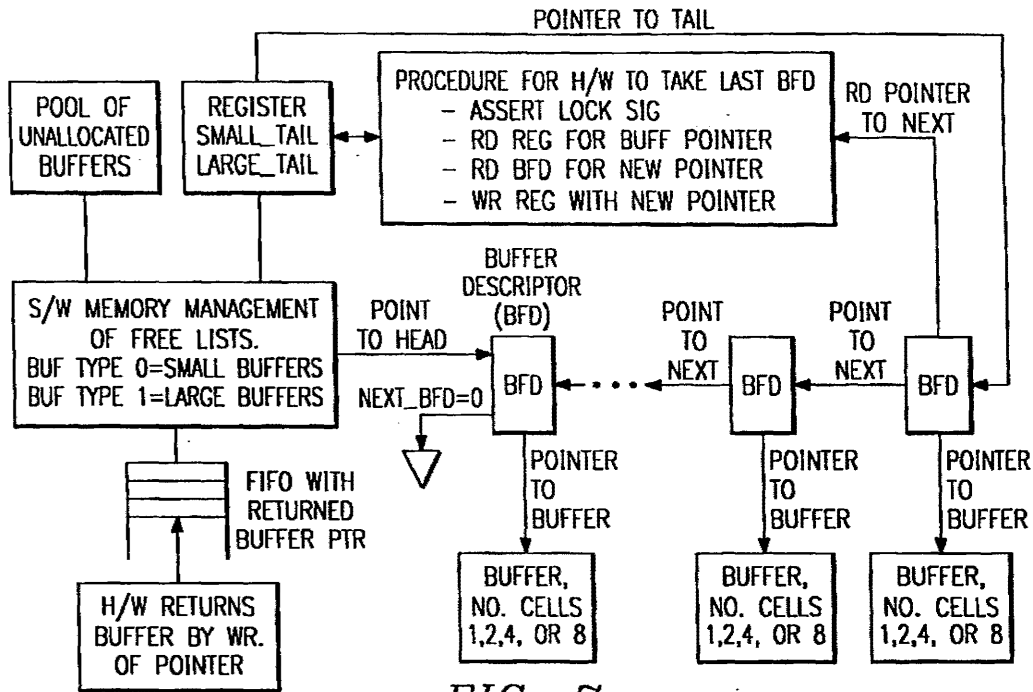


FIG. 7

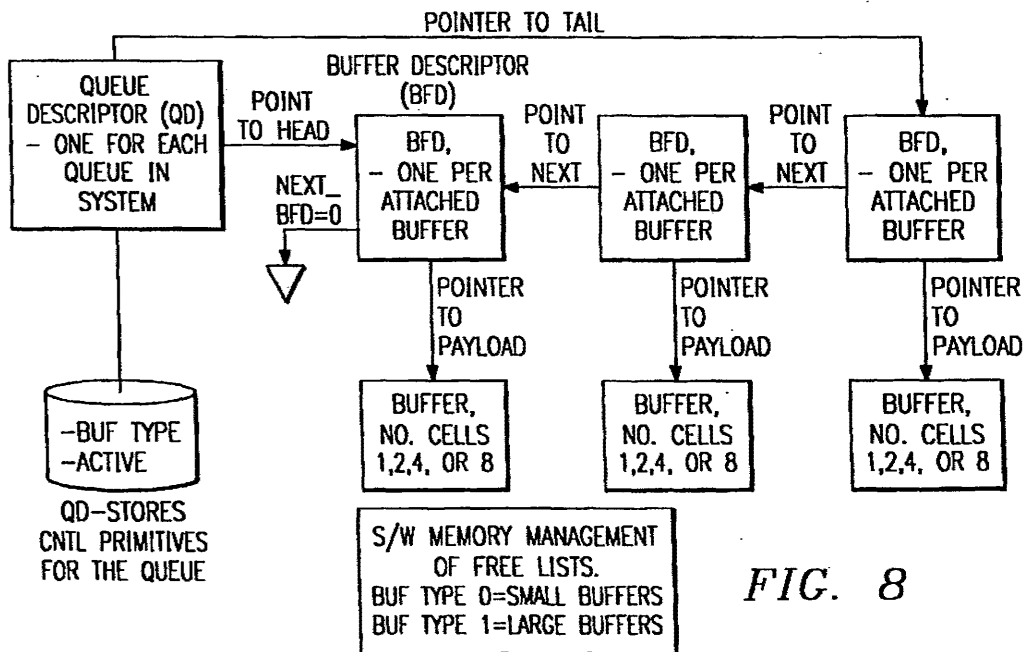


FIG. 8

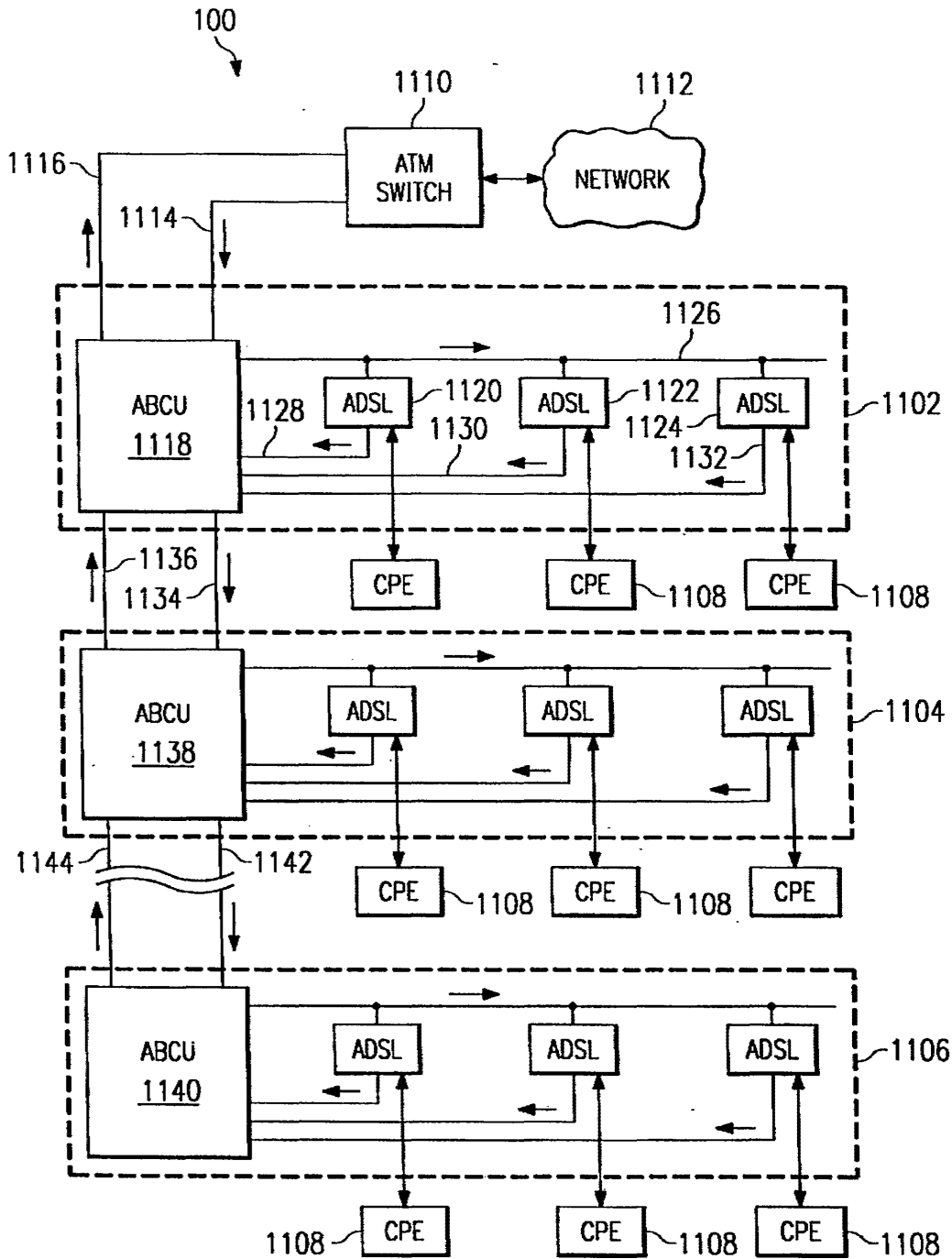
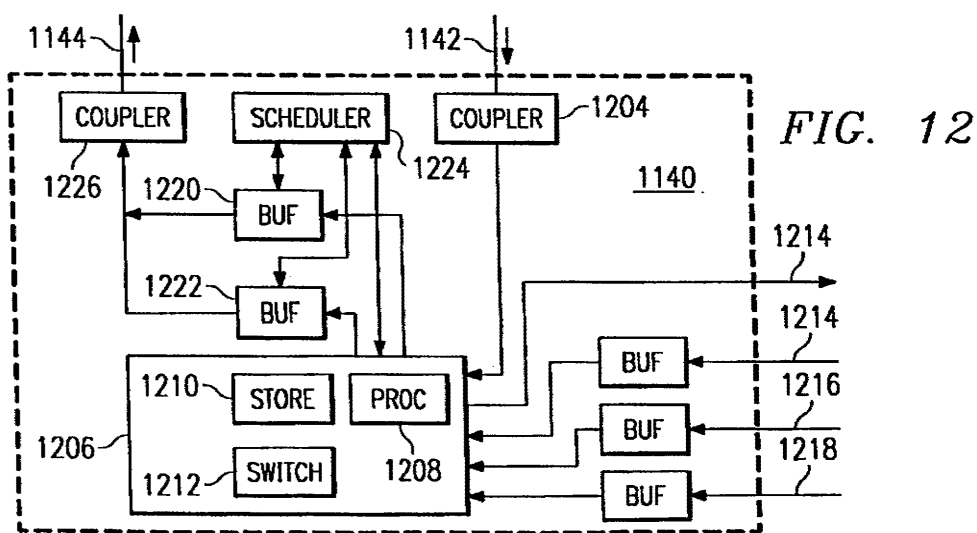
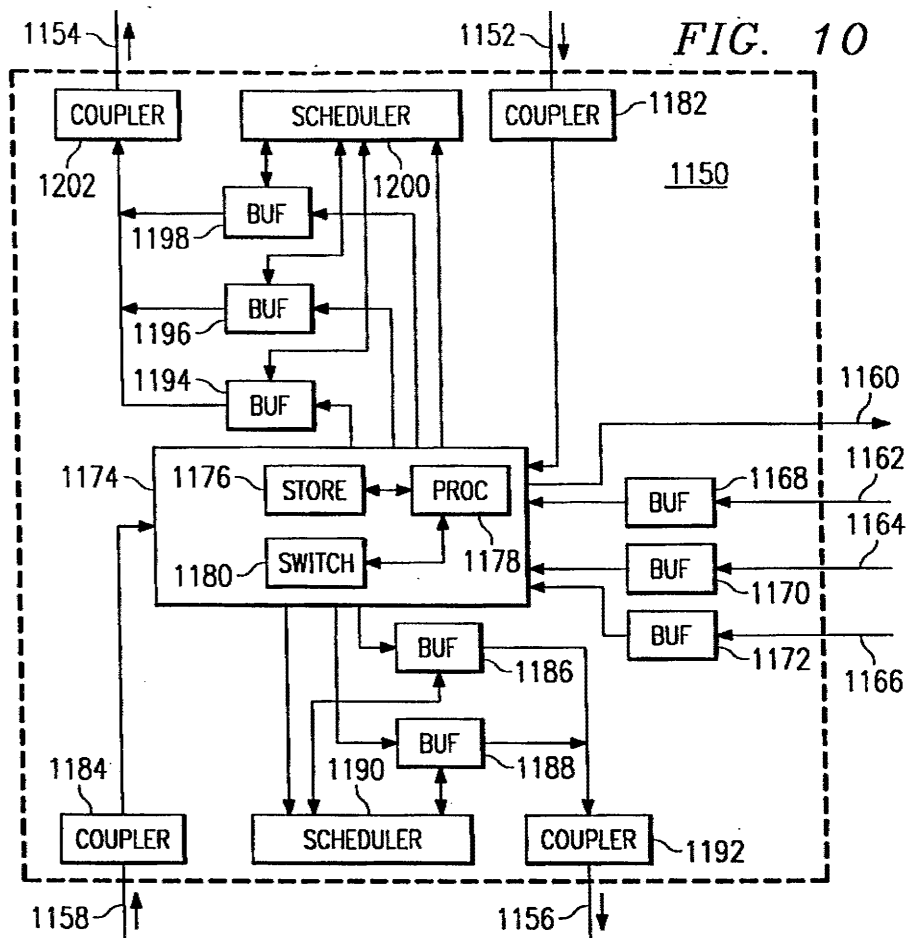


FIG. 9



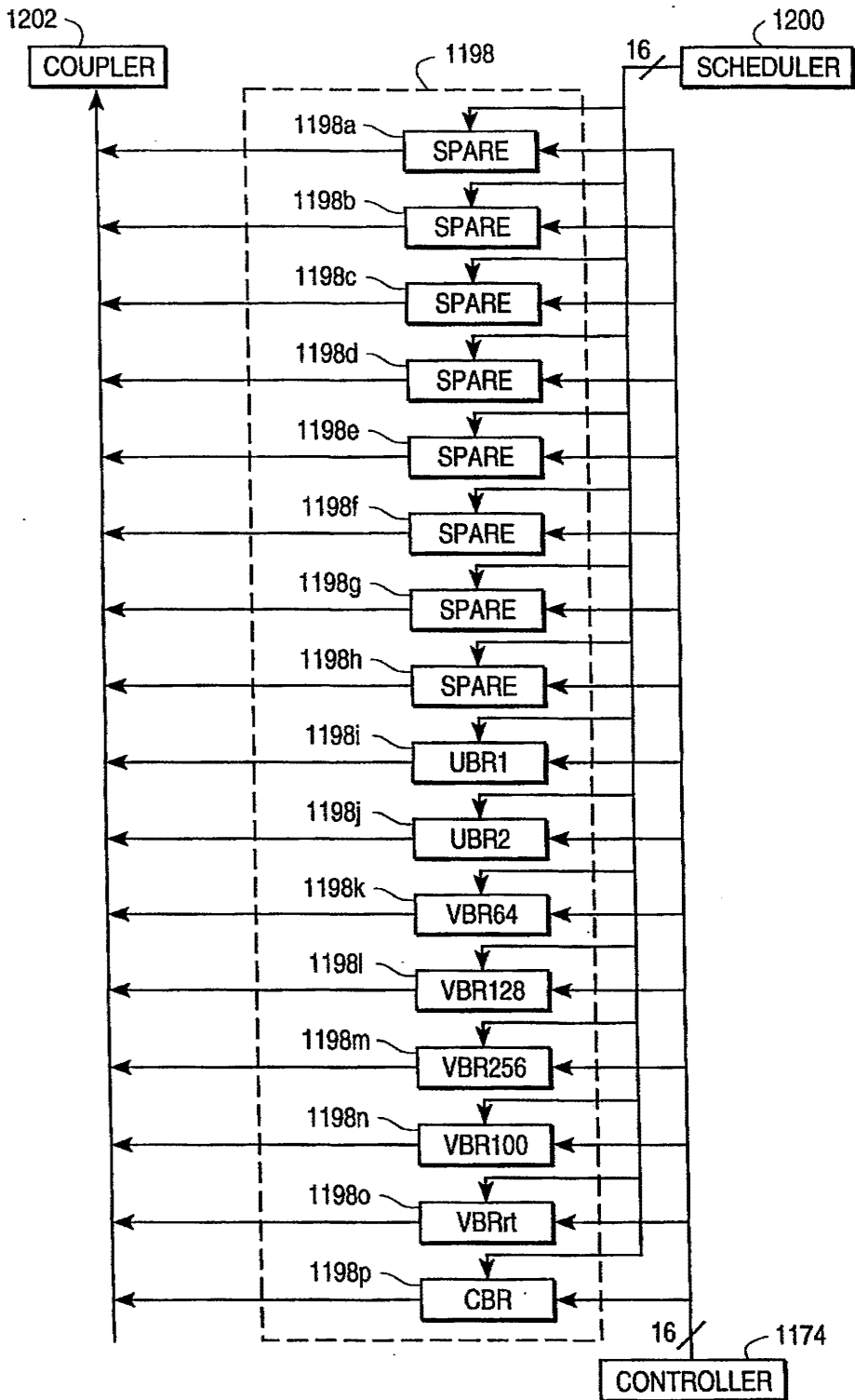


FIG. 11

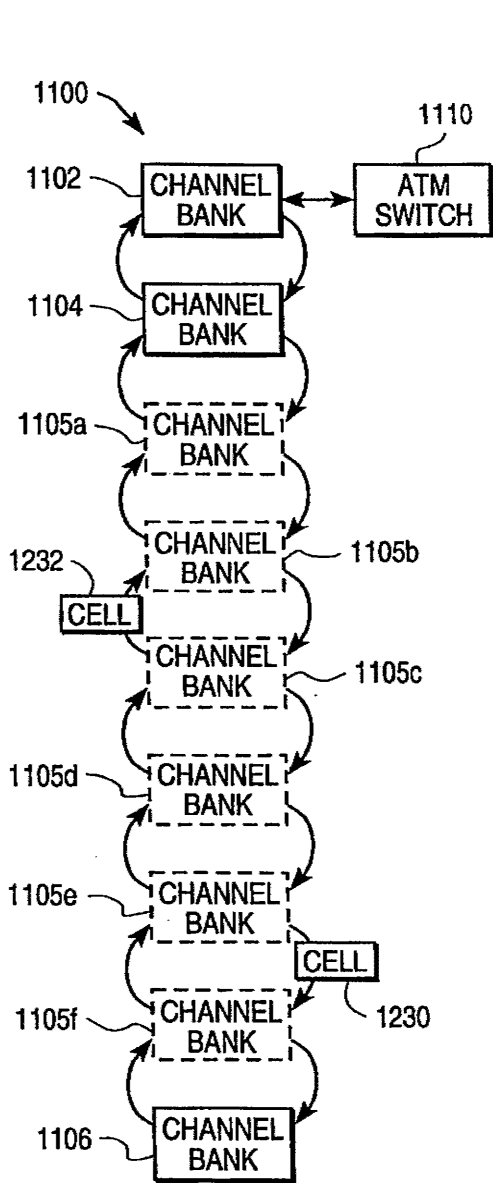


FIG. 13

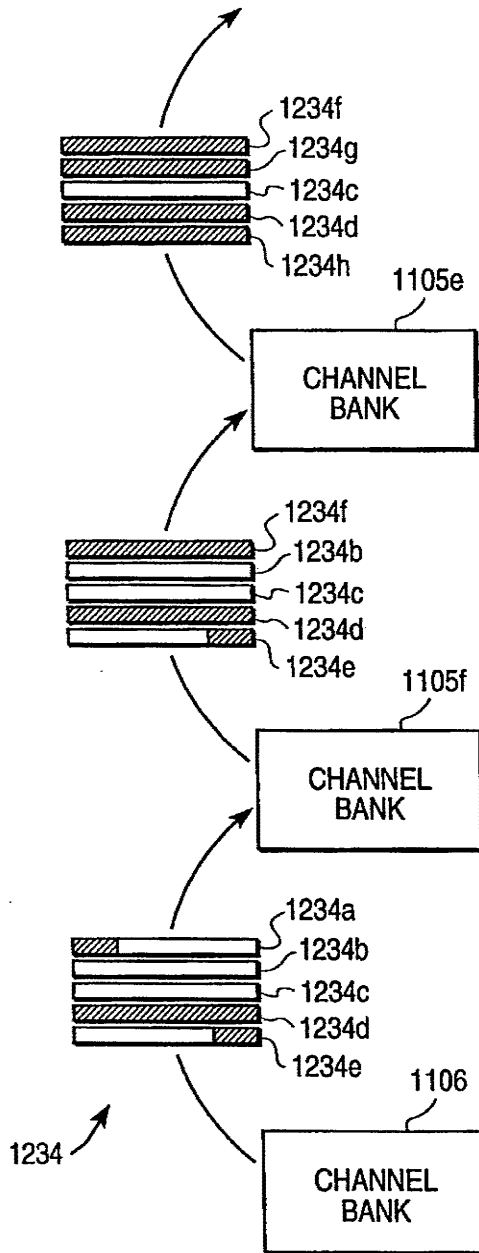


FIG. 14

DISTRIBUTED TELECOMMUNICATIONS SWITCHING SYSTEM AND METHOD

This utility patent application claims the benefit of U.S. Provisional Application Ser. No. 60/032,609, entitled "Technique and System for Accessing Asynchronous Transfer Mode Networks," filed on Dec. 4, 1996.

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to the field of telecommunications switching and more particularly to a distributed telecommunications switching system and method.

BACKGROUND OF THE INVENTION

A variety of telecommunications networks have been used to establish communication between customer premises equipment (CPE) units and a central office. Most of these networks are formed in a "tree" structure, in which the central office is connected to several switching units, which are each connected to several smaller switching units, and so on along the "branches" of the tree. At the lowest level of switching units, each unit is connected to one or more CPE units.

To route addressed data or otherwise communicate with one of the CPE units, the central office determines which branch of the tree services the CPE unit in question. The data is then passed to the switching system for that branch, which in turn passes the data on to the next lower level in the switching hierarchy, and so on, until the data reaches the CPE unit.

This routing scheme requires that each switching system at each level in the hierarchy must store address and routing information for all of the CPE units serviced by it. If the customer base is expanded to include additional CPE units, then all switching systems routing traffic to the new CPE units must be reprogrammed to store the new address and routing information. Therefore, it is desirable to avoid establishing, maintaining, and updating address and routing information storage for the entire network at each switching system therein.

SUMMARY OF THE INVENTION

Therefore, a need has arisen for a telecommunications switching system that addresses the disadvantages and deficiencies of the prior art.

A distributed telecommunications switching system is disclosed. The system includes a controller that generates and transmits a control signal which carries a plurality of credit allowance values. A first switching system receives the control signal and transmits data packets to a second switching system in response to a first credit allowance value. A third switching system receives the control signal and transmits data packets to the first switching system in response to a second credit allowance value.

A technical advantage of the present invention is that subscribers to over-subscribed traffic classes receive service mediated by an upstream flow control process. A further advantage is that upstream bandwidth is distributed to subscribers in fair manner independent of the switching system to which a subscriber is connected.

The present invention may be advantageously used to facilitate access to asynchronous transfer mode ("ATM") networks and environments.

The present invention provides for a technique and system that can be employed to interface with, and provide access

to, an ATM network. The present invention may be employed to interface with an ATM network, such as central offices of a public switched telephone network or wide area networks that operate through the transmission of optical signals in ATM format, and to route information between the ATM network and designated subscriber interfaces. For example, the present invention may be interposed between a wide area network having an ATM backbone and customer premise equipment. Such placement allows for the present invention to provide different functions on behalf of the wide area network (such as a "policing" function, which regulates the traffic flow to wide area networks), as well as on behalf of the customer premise equipment (such as a rate adoption function for local area networks).

Multiple interconnected units (which can also be referred to as "shelves" or "channel banks") are preferably used to implement the present invention. The multiple units may be physically located in a common place or in remote locations from one another. Each unit is associated with a plurality of subscriber interfaces, and performs distinct functions and procedures to the traffic deriving from the ATM network or subscriber interfaces. The cumulative effect of the multiple units is to form a technique and system that, among other things, routes and controls the ATM traffic amongst the various subscriber interfaces. As such, the present invention can be considered as a series of distributed ATM switches or nodes that collectively function as a single switching or multiplexing entity.

Preferably, the units are serially connected to one another (i.e., daisy-chained) such that any one unit is connected to one or two other units. The first and last units are connected to only one other unit, while the intermediate units between the first and last units are connected to two other units.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying drawings, wherein like reference numerals represent like parts, in which:

FIG. 1 illustrates a block diagram of a telecommunications network;

FIG. 2 illustrates a block diagram of a portion of a switching subsystem within the telecommunications network;

FIG. 3 illustrates a block diagram of a channel bank unit within the switching subsystem;

FIG. 4 illustrates another block diagram of the channel bank;

FIG. 5 illustrates a block diagram of a top level memory fabric of the switching subsystem;

FIG. 6 illustrates a block diagram of the fabric controls of the switching subsystem;

FIG. 7 illustrates a block diagram of the memory management within the switching subsystem;

FIG. 8 illustrates a block diagram of a logical queue structure within the switching subsystem;

FIG. 9 is a block diagram of a distributed telecommunications switching subsystem;

FIG. 10 is a block diagram of a controller for use in the distributed switching subsystem;

FIG. 11 is an expanded block diagram of an ingress queue system for use in the distributed switching subsystem;

FIG. 12 is a block diagram of a terminating controller for use in the distributed switching subsystem;

3

FIG. 13 is a block diagram illustrating a first upstream flow control system for the distributed switching subsystem; and

FIG. 14 is a block diagram illustrating a second upstream flow control system for the distributed switching subsystem. 5

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 is a block diagram of a telecommunications network 10. Telecommunications network 10 includes a central office 11, an asynchronous transfer mode (ATM) switch 12, a time division multiplex (TDM) switch 13, a central digital loop carrier system 14, a remote digital loop carrier system 15, one or more remote terminal 16, and a switching subsystem 100. In operation, central office 11 may receive time division multiplex traffic from TDM switch 13 at any of a plurality of channel banks 17. The TDM traffic is received by a line card 18, appropriately processed, and transferred to a common control shelf 19 where the TDM traffic can be passed to an appropriate central digital loop carrier system 14. Digital loop carrier system 14 may also receive ATM traffic from ATM switch 12. Digital loop carrier system 14 integrates the TDM traffic with the ATM traffic for transfer, preferably over an optical fiber link 20 to a remote digital loop carrier system 15. The remote digital loop carrier system 15 may partition the integrated TDM and ATM traffic received over optical fiber link 20 into separate TDM and ATM traffic streams. The partitioned TDM traffic stream may be provided to a remote terminal 16 according to its appropriate destination. Digital loop carrier system 15 may also provide the partitioned ATM stream to switching system 100 that appropriately sends the ATM stream to its appropriate user destination. While FIG. 1 shows switching subsystem 100 as only receiving an ATM stream, switching subsystem 100 may also receive and process TDM streams from telecommunications network 10.

FIG. 1 illustrates a switching subsystem 100, also known as an ATM service access multiplexer, in accordance with a preferred embodiment of the present invention. As illustrated, switching subsystem 100 may communicate to an ATM switch 12, as commonly found in a central office 14, through a shared communication link 110. Switching subsystem 100 includes a first switching unit 104a, a last switching unit 104n, and one or more intermediate switching units 104i interposed between the first switching unit 104a and the last switching unit 104n. Such switching units 104 are connected to one another through bidirectional connections 108, which collectively can be considered to provide for a control loop 112. Such connections 108 preferably transmit optical signals, such as OC-3 optical signals. Further, the switching units 104 are each associated to certain subscriber interfaces, specifically, the first switching unit 104a is associated with certain subscriber interfaces by link 106a, the intermediate switching units 104i are associated with other subscriber interfaces by link 106i, and the last switching unit 104n is associated with still other subscriber interfaces by link 106n.

Bi-directional connections 108 between the units allow for the transmission of control and routing information to be transferred between the units. The transmission of information may be in the downstream direction, such as from the first switching unit 104a to the intermediate switching unit 104i that it is directly connected to. Similarly, information may be transmitted in the upstream direction, such as from the last switching unit 104n to the intermediate switching unit 104i that it is directly connected to. Various levels of

4

control may be provided by the switching subsystem 100, such as the following:

- (1) instantaneous controls, including controlling the routing of ATM cells, the discarding of selective ATM cells, the signaling of ATM cell mapping, statistics gathering concerning ATM cells, and the marking of ATM cells;
- (2) real time controls, including controlling the management of ATM cell buffers, the analysis and assessment of "fairness" for various classes of service, and the computation of queue occupancies;
- (3) hop-by-hop (or segment-to-segment) propagation delay controls;
- (4) end-to-end propagation delay controls; and
- (5) end-to-end round trip delay controls.

ATM cells include information such as virtual path ("VP") and virtual circuit ("VC") routing information, and information concerning their termination ("terminating information"). Each switching unit 104 analyzes and evaluates the information included with each ATM cell. If the ATM cell identifies VP or VC routing information that is associated with a particular switching unit 104 analyzing the cell, then the cell is forwarded by that particular switching unit to the appropriate destination. Similarly, if the ATM cell includes terminating information that is associated with the particular switching unit 104 evaluating the cell, then the cell is terminated by that particular switching unit 104. In the absence of matching routing or terminating information between the ATM cell and the evaluating switching unit 104, then the evaluating unit passes the cell downstream to the next switching unit 104. That switching unit 104 will then undertake similar analyses and evaluations. As a result, certain switching units 104 are operable to forward or terminate certain ATM cells. However, when the multiple switching units 104 are considered collectively, they are able to either forward or terminate all of the ATM cells. As such, the switching subsystem 100 provides for a distributed switching technique.

A conformant stream of information is preferably transmitted between the switching units 104. Such conformant stream is established by the imposition of control procedures through the use of the control loop 112.

A fairness analysis and credit-based scheme may, for example, be established through the control loop 112 to control upstream congestion between the switching units 104. The first switching unit 104a preferably generates a command cell in the downstream direction. The command cell includes information that defines the credits to be awarded to each unit switching 104, in accordance with the fairness analysis and assessment, and effects the downstream serial transmission of that cell to the other units. In response to reception of the command cell, the last switching unit 104n generates a feedback status cell, which includes feedback status information, such as the congestion status and behavioral attributes of a given shelf. The feedback status cell is, however, passed upstream and the feedback information therein is modified by the intermediate switching units 104i. Specifically, each intermediate switching unit 104i preferably supplements the information already included in the feedback status cell, which concerns other units, with feedback information concerning that particular unit. Using the information provided for in the command cell, together with the feedback status cell, allows for a credit-based scheme to take place whereby each switching unit 104 is informed of the number of credits it is awarded. The number of credits relates to the number of ATM cells that a switching unit 104 can pass upstream in a given period

5

of time. Upon receiving the credits, a particular switching unit 104 may start to launch ATM cells into the upstream connection 108 until its credits are exhausted. The above-described fairness analysis and credit-based scheme is preferably implemented by designating one of the switching units 104 as a master, and the other units as slaves. The master switching unit 104, preferably the first switching unit 104a, should be operable to compute the credits awarded to each slave unit switching 104 based on the command and feedback status cells, and to inform each slave switching unit of its allotted number of credits. As a consequence of the fairness analysis and credit based scheme, the connections 108 between the switching units 104 are regulated such that upstream congestion (i.e., a bottleneck) is avoided.

FIG. 2 is a block diagram of switching unit 104. Switching unit 104 includes an asynchronous transfer mode bank channel unit (ABCU) card 22 and a plurality of asynchronous digital subscriber line (ADSL) line cards 24. While ADSL line cards 24 are to be described preferably with respect to the asynchronous digital subscriber loop protocol, ADSL line cards 24 may be implemented with other appropriate transmission protocols. In operation, switching unit 104 receives asynchronous transfer mode cells at an ATM cell multiplexer add/drop unit 25. Add/drop unit 25 determines whether each ATM cell received has the appropriate destination and addressing information for a user serviced by ADSL line cards 24 associated with ABCU card 22. If not, then the ATM cell is passed to the next switching unit 104 within switching subsystem 100. If add/drop unit 25 identifies an ATM cell with the correct addressing and destination information, then the ATM cell is forwarded to an appropriate bus interface 26 for transfer to an appropriate ADSL line card 24. The appropriate ADSL line card includes a bus interface 27 to extract the ATM cell and provide it to a transceiver 28 where the ATM cell is placed into the appropriate ADSL transmission format for transmission to a remote unit 29. Remote unit 29 processes the ADSL transmission received from ADSL line card 24 through a transceiver 30, physical layer unit 31, segmentation and resegmentation unit 32 or other appropriate device and a user interface 33 for transmission to an end user.

ABCU card 22 may receive TDM traffic over a timeslot interchange cable 34 from TDM switch 13 through a switching device such as digital loop carrier system 15. ABCU card 22 includes a timeslot assigner 35 that places the TDM traffic into a subscriber bus interface (SBI) protocol format. The TDM traffic in the SBI protocol format is provided to an SBI selector 36 and sent to the appropriate ADSL line card 24 for transmission to the end user.

In the upstream direction, ADSL line card 24 receives an ADSL transmission from remote unit 29 and places the ADSL transmission into an appropriate ATM or TDM traffic stream at bus interface 27. The ATM and TDM traffic streams are transferred to a corresponding SBI selector 36 in order to provide the TDM traffic to timeslot assignment 35 and the ATM traffic to add/drop unit 25.

FIG. 3 is a simplified block diagram of ABCU card 22. In the downstream direction, ABCU card 22 receives asynchronous transfer mode cells from ATM switch 12 at a physical layer interface 40. A downstream virtual path (VP) lookup table 42 and a downstream virtual circuit (VC) lookup table 44 are used in determining whether the ATM cell is destined for this ABCU card 22. A comparison is done at downstream VP lookup table 42 to determine whether there is a match in the VP addressing. If a match occurs, the ATM cell is placed into an appropriate queue 46 and is scheduled for transmission to associated ADSL line card 24.

6

If a match did not occur at downstream VP lookup table 42, a comparison is done at downstream VC lookup table 44. If a match occurs at downstream VC lookup table 44, then the ATM cell is sent to queue 46. If a match still has not occurred, a downstream CPU lookup table 48 is consulted to determine if the ATM cell is a control cell to be processed by the CPU on the ABCU card 22. If a match occurs at the downstream CPU lookup table 48, the ATM cell is passed to the CPU of ABCU card 22. If there is still no match, then the ATM cell is not destined for this ABCU card 22. The ATM cell is then passed to the next switching unit 104 within switching subsystem 100. The next switching unit 104 performs a similar lookup process described above. ATM cells provided to ADSL line card 24 are placed into a buffer 50, processed by a transmission convergence layer 52, and sent to the remote unit 29 through a physical layer interface 54.

In the upstream direction, ADSL line card 24 receives an ADSL transmission from remote unit 29 and physical layer interface 54. The ADSL transmission is processed by TC layer 52 and the resulting traffic is placed into a buffer 56. The resulting traffic is sent from buffer 56 to a holding queue 58 on ABCU card 22. Comparisons are done on the traffic cell at upstream VP lookup table 60 and upstream VC lookup table 62. If no match is found, the traffic cell is sent to the CPU of ABCU card 22 for further processing. If an appropriate match occurs, the traffic cell is placed into an upstream queue 64 where it awaits scheduling for transmission by a credit scheduler 66.

ABCU card 22 also receives ATM cells from another switching unit 104. ATM cells received from another switching unit 104 are processed by an upstream CPU lookup table 68 determined whether the receive cell is a control cell. If so, the ATM cell received from another switching unit 104 is passed to the CPU of ABCU card 22 for further processing. If it is not a control cell, the ATM cell received from another switching unit 104 is placed into a hop queue 70. A selector 72 determines which of the cells in the hop queue 70 and the cells identified by the credit scheduler 66 from the upstream queue 64 are to be transmitted through a physical layer interface 74 to ATM switch 12.

FIG. 4 provides a more detailed view of ABCU card 22. ABCU card 22 may have a switch controller 80 that performs the VP and VC lookup of ATM cells received from ATM switch 12. If a VC or VP match occurs a controller 80, the ATM cells passed to queue 46 are sent to an appropriate ADSL line card 24 as determined by a scheduler 82. A rate adapter transfers the ATM cell from queue 46 over the cell bus to ADSL line card 24 at the appropriate rate as determined by an adapter controller 86. If a lookup match does not occur at controller 80, the ATM cells are placed into a CPU queue if determined to be a control cell, or a bypass queue area 88 for transfer to the next switching unit 104. Transfer from the bypass/CPU queue 88 is determined by a scheduler 90. ATM cells within bypass queue 88 are transferred to a TC layer 92 and a physical layer interface 93 to the next switching unit 104.

Cells received from another switching unit 104 through physical layer interface 93 and TC layer 92 are processed by switch controller 80. Switch controller 80 identifies the destination for the ATM cell received from switching unit 104 and places it into the appropriate queue area 94 for external transfer or one of the other queues for internal transfer. Switch controller 80 also may receive cells from ADSL line cards 24 through buffer unit 58 as loaded by a recovery unit 96. Cells not in an appropriate ATM cell format are placed into a queue 97 and processed into the

proper format by a segmentation and resegmentation (SAR) unit 98. Other types of processing may be performed on ATM cells analyzed by switch controller 80 through a per VC accounting unit 81, a cell discard unit 83, and a usage parameter control policing unit 85. Switch controller 80 may also interface with TDM traffic received from TSI cable 34 through a custom TC layer 87 and a physical layer interface 89. Switch controller 80 may also provide cells for conversion to TDM format to a buffer queue 75. Cells in buffer queue 75 are transferred to a TDM cell layer 77 as determined by a scheduler 79 and then sent over TSI cable 34 through a physical layer interface 73. Switch controller 80 is capable of processing ATM cells and TDM traffic for internal and/or external routing and rerouting of traffic from any place of origination to any place of destination.

A. OVERVIEW

Switching subsystem 100 provides daisy chaining of multiple units or shelves. The present invention is described as having nine switching units concurrently, however, any number of intermediate shelves 104_i may be implemented. In other words, all the switching units cooperate to implement a distributed switch process and distributed real time control processes including fairness. The daisy chaining queue (bypass) method takes priority over all local queues. In effect each switching unit generates a conformant cell stream where the sum of all the rates from the multiple switching units equal the OC-3 rate. The resultant behavior of the nine switching units is equivalent to a single ATM switching node.

Switching subsystem 100 implements advanced functions that are generally transparent to the data path traffic. A control loop 112 between the switching units 104 permits the switching units 104 to cooperate on the various system functions. The switching units 104 are classified as first, intermediate or last. The first, intermediate, and last switching units 104 generally run similar software, but each of the switching units 104 may have its own unique processes. Switching subsystem 100 is capable of both VP and VC cell routing with support for up to eight or more traffic classes. The control levels provided by the switching subsystem 100 can be grouped into five categories, although other control levels are possible. The five exemplary categories are:

- 1 instantaneous controls
 - cell routing
 - selective cell discard (EPD, PPD)
 - signaling cell mapping
 - cell statistics gathering
 - EFCI/CLP marking
- 2 real time controls
 - control process for cell buffer management (to declare congestion states)
 - compute fairness primitives (i.e. EPD rates)
 - compute queue occupancy
- 3 hop-by-hop propagation delay controls (or segment-to-segment)
 - inter-shelf peer to peer element state signaling (i.e. for fairness process)
- 4 end-to-end propagation delay controls
 - EFCI flow control
- 5 end-to-end round trip delay controls
 - CAC I/F via NMS
 - routing provisioning via NMS

Switching units 104 cooperate to implement a distributed ATM switch. These switching units 104 can be either co-located or remotely dispersed.

These switching units 104, preferably nine (9) in number, cooperate to implement a single ATM switch node. Different procedures are required in the three types of shelves (first, intermediate, last) to implement the distributed switch functions.

In one embodiment, an asynchronous transfer mode bank channel unit (ABCU) card 22 resident within each switching unit provides the functionality for the distributed MegaSLAM switch. The nine switching units 104 are daisy chained via their corresponding ABCU cards and may reside in different locations.

The logic resident on the ABCU card 22 implements the cell routing function for any ingress cells from either the network OC-3c, the daisy chain OC-3c or the Upstream time division multiplexed (TDM) bus stream. The virtual circuit validation process is a two stage process.

The first stage logic on the ABCU card 22 checks to see if a virtual path (VP) connection is provisioned for this ingress cell. Each ingress interface can be provisioned to support either user to network interface (UNI) or network to network (NNI) interfaces. The virtual path lookup is preferably a linear table where the 8/12 VP bits point to a VP_descriptor. Thus, a table with 256 byte or 4 Kbytes VP_descriptor entries may be used. The VP_descriptor contains the required connection information. If the virtual path lookup is successful, then the cell level processing is implemented by the ABCU card 22 and the cell is forwarded to the appropriate subscriber interface destination. Use of the linear lookup provides for a fast return of a VP_lookup_failure indication in the event of a virtual path look up failure. Preferably this indication will be provided to the next stage within two clock cycles.

A virtual circuit (VC) lookup sequence is triggered by the VP_lookup_failure indication from the previous state. The virtual circuit lookup is preferably implemented in hardware by a sorted list that supports a maximum of 2K virtual paths. The process starts near the middle of the list and tests to see if the current 24/28 bit virtual circuit bit pattern is equal to, greater than or less than the pattern from the VP_descriptor entry. This hardware test preferably requires 2 clock cycles, or less, to complete. At 50 MHZ, this would permit 25 iterations each requiring 40 ns before the 1.0 μ s deadline. For a VC range that is a power of 2, the number of iterations is equal to the exponent plus one (2^{11} supports 2000 virtual circuits which requires $11+1=12$ iterations) This design, whether implemented in software, firmware, or an Application-Specific Integrated Circuit (ASIC) may be used in OC-3, OC-12, or other applications. This design further may support applications having 64,000 virtual circuits or more.

If the above two checks failed, then the cell is tested by a third stage that evaluates the cell against 8 registers sets that preferably identify CPU terminated cells. If a match condition is found, the cell is passed to the ABCU card 22 resident CPU. These registers can be programmed to strip operation, administration, and maintenance (OAM) cell, resource management(RM) cells, and other cells, out of the cell streams.

In the event all three lookups fail for the current cell, the cell may be passed via a separate FIFO to the next switching unit 104 in the daisy chain. This permits the switching units 104 to implement the distributed switch process. In effect, each switching unit 104 is programmed to terminate a subset of the virtual circuits for the downstream path. But as a whole all the switching units 104 terminate all the downstream virtual circuits. The last switching unit 104_n implements the mis-inserted cell processing function as a proxy

for all the daisy chained switching units 104. Thus, the last switching unit 104 n acts as a proxy for all exception events for the distributed switch fabric.

The nine distributed switching units 104 cooperate to produce a conformant stream that not only fits into the communication link bandwidth, but at the same time provides fairness to the class of service that is oversubscribing the shared communication link to the CO resident ATM switch. A control loop initiated by the first switching unit 104 a preferably provides the primitives necessary to implement the distributed fairness process. The feedback control loop is initiated by the last switching unit 104 n and the cell is modified by the intermediate switching units 104. A credit based scheme is implemented and the first switching unit 104 a tells all other switching units 104 how many cells they can send for the given control period. The fairness analysis in the first switching unit 104 a is undertaken to compute the credits that each switching unit 104 gets for a given control period.

The fairness algorithm will generate conformant streams in each switching unit 104. Each switching unit 104 in the daisy chain treats the upstream daisy chain as the highest priority stream and queues the cell in the Bypass queue. The locally conformant stream is derived from the output side of the `Ingress_queue_[7 . . . 0]`. The queues are serviced on a priority basis with a credit based algorithm. The logic on the ABCU card 22 generates the conformant stream by launching the permitted number of cells during the current control period. Assuming the control period is in fact equal to 128 cell times on the OC-3c, then each switching unit 104 is permitted to launch its portion of the 128 cell budget. The credit based scheme guarantees that the physical OC-3 pipe never becomes a bottleneck in any of the daisy chained links.

The fairness algorithm, and its associated credit based control function, for the multiple switching units 104 should be based on a control interval fast enough such that the ingress cell exposure does not consume more than a small fraction of the total buffer resources (say 5% max). It is believed that a stable (non-oscillating) algorithm is possible if the rate of change of the aggregate cell buffers is limited to a small number <5%. The planned aggregate cell buffer is 8K cells. Thus, five percent exposure would be about 400 cells. If the ingress rate is worst case 1.0 us per cell then the control process should be faster than 400 us.

The basic mechanism that permits that upstream algorithm to operate in a distributed manner over the daisy chained switching units 104 is the credit based scheduler in each switching unit 104. The credit based scheduler cooperates with the controller in the first switching unit 104 a . The communication of the controlling primitives is accomplished with out of band control cells. One point to multi-point cell is sent in the down stream direction from the first switching unit 104 a of all subordinate switching units 104. This downstream cell contains the primitive that defines the credit granted to each switching unit 104. In response to this downstream control cell the last switching unit 104 n initiates a feedback status cell that each switching unit 104 modifies which is eventually terminated on the first switching unit 104 a . The feedback cell contains one or more primitives that define the congestion status and or queue behavioral attributes of the given switching unit 104.

The upstream buffer resources are organized into a free list of buffers. The size of the buffers is a provisioned parameter but during system run time one fixed size may be used is 64 byte aligned. The size may be 64, 128, 256 or 512 bytes. The cells are mapped into the buffers as 52 bytes. The

free list of buffers has three trigger levels plus one normal level they are;

Congestion Level	Level Intent	Functions
Level zero (L0)	Normal state	All cell streams are queued and forwarded to target spots
Level one (L1)	Trigger status signaling	CLP marking EFPI marking Future ABR procedures or credit based flow control procedures
Level two (L2)	Congestion Imminent	discards policies on a selective basis -early packet discard -partial packet discard -fairness algorithm with per class or per group granularity Future enhancements per class or per group differentiated procedures
Level three (L3)	Congestion	aggressive discard policies -cell level discards per group or class granularity Goal: at all cost protect the highest priority Qos guaranteed streams.

If no levels are triggered (i.e. level zero), then all ingress cells are enqueued in the eight queues as a function of the `VC_descriptor` parameter. The eight queues can be serviced with any algorithm with one being a priority algorithm. The cells are then mapped into the OC-3 PHY layer. If level one is triggered, then CLP marking and EFPI marking is implemented on the programmed number of cell streams destined to some of the queues. If level two is also triggered, then level one procedures remain in effect. This is possible because packet level discard will occur before the cells are queued into the respective queue. The EPD procedure operates on ingress cells with port granularity. The total number of EPD circuits implemented are shared among the ingress ports. Each ingress cell is associated with a `VC_descriptor` and the target queue is defined in the `VC_descriptor`. The aggregate of all upstream VCI/VPI are evaluated against the active EPD logic elements that are shared with all the ports. These EPD logic elements store the context of the in-progress packet discards. If there is a match, then the EPD or PPD procedure is implemented by the hardware. In other words the cell is not queued in one of the 8 queues. A pipelined implementation is envisioned where the `VC_descriptor` lookup occurs and a primitive is appended to identify the target queue and source port. The next state in the pipeline evaluates the cell to match it for a discard VCI/VPI in progress for the given port. This means TBD packets destined for one of eight queues can all be in the discard mode until the end of message (EOM) marker state. The action of writing the `EPD_cntl[]` register sets a go command flag. The initialization of the `EPD_cntl[]` registers is implemented by a write cycle to the register.

The key item here is that each switching unit 104 manages its own congestion state and discard procedures to enforce fairness. Any locally computed status primitives can be encoded and placed into the upstream status cell that is part of the control loop.

The 10 upstream queues are serviced by a controller that launches a predetermined number of cells during the current control period. The upstream controller for the outbound OC-3c services 2 of 10 queues using a priority algorithm while the remaining 8 queues can use a locally defined

algorithm. The two queues serviced with the priority algorithm are the Bypass queue and the CPU queue. Each queue is serviced by a scheduler and one provided scheme may be to read each queue until empty before advancing to the next queue. The controller blindly launches all cells from the Bypass queue and the CPU queue since it is assumed that these streams are already conformant and have been previously scheduled by another shelf. The CPU cells are important for real time controls but are considered negligible from a system load point of view. The cells from these two queues are not counted by the controller. The controller is granted a fixed number of credits for the local ingress queue [7 . . . 0] for the current control period. As the controller services these queues, the credit counter is decremented until it reaches zero. At this point, the controller stops and waits for the next control period before launching any more cells. Due to boundary conditions the controller may not reach zero before the end of the control period. The controller, when reinitialized for the next control period, remembers the remainder from the previous period. The controller, during the current period, may first exhaust the counter from the previous period before decrementing the counter for the current period.

The boundary conditions impact the accuracy of the fairness algorithm. It is expected that the delay of remote daisy chained switching units 104 may cause short term bursts from these switching units 104 that appear to be in excess of the allocated credits.

The deadline for feed time controls are about two or three orders of magnitude slower than the per cell deadline. These controls are all implemented by a RISC CPU on the ABCU. The CPU is expected to cooperate with the peer CPUs in other shelves that may exist in a daisy chained configuration.

In the downstream direction, the cells are fanned out to their target switching units 104 via the VC/VP descriptor lookup in each switching unit. In the VC case, the cells are enqueued into either a high priority or a low priority queue that is associated with each drop (or port). The ASCU card 22 is capable of 22 sets of these dual priority queues.

Each queue uses a real time buffer attached to the queue from the free list.

When the downstream direction is in the L0 congestion mode, then all queues get whatever buffer attachments they want.

When the downstream direction is in the L1 congestion mode, then the cells are conditionally EFCI marked and some low priority traffic classes may be CLP marked.

When the downstream direction is in the L2 congestion mode, then a pool of PPD engines are invoked and the controlling software is required to drive these discard engines to fairly discard between all the active low priority queues in the system.

When the downstream direction is in the L3 congestion mode, all cells going to the low priority queue are discarded in all switching units 104.

The process of mapping cells over the shared downstream cell bus is implemented with a provisioned rate adaptation procedures. Feedback over the TDM bus provides the mechanism to ensure that the small FIFO on the line channel card 24 does not overflow or underflow.

Each switching unit 104, on its own initiative, implements the congestion policies and thus each switching unit 104 may be at a different congestion level. It is felt that if sufficient buffer resources are allocated to the downstream path, then interference generated by the upstream path consuming buffer resources can be minimal.

All the slave switching units 104 participate in generating a feedback status cell that is sent to the first switching unit

104a. This cell contains the congestion state, the free list size and future primitives for the downstream direction.

Two types of control cells exist one initiated by the first switching unit 104a and sent to all daisy chained switching units 104 and another generated by the slave switching units 104 and terminated on the first switching unit 104a.

Master generated control cell as mapped into OAM format;

Octet	Function
1 .. 5	standard ATM header
6	-4 bits OAM type -4 bits Function type coding TBD
7 .. 8	Control command word, -TBD many contain length of control cycle in cell times etc.
9 .. 24	credit_cntl [7.0] 8 words of 16 bits contain the credit allowance for each of the 8 daisy chained shelves. octets #9&10 are for the first subordinate shelf etc. octets #23&24 are for the last shelf
25 .. 26	spare - for future primitives
47-48	-6 bits reserved -10 bits for CRC-10

The 16 bit control word for each of the slave switching units 104 has the following format, i.e., credit_cntl[7 . . . 0]

Bit	Function
0 .. 9	number of cell granularity credits granted by master shelf
10 .. 15	reserved for future primitives;

The first switching unit 104a runs an algorithm that computes the credits as a proxy for all first switching units 104. The first switching unit 104a operates on a fixed control period. A reasonable fixed period may be 128 cell time intervals on an OC-3 link. This period is about 350 μs. During this time, the first switching unit 104a computes the credits for each of the other switching units 104a. The sum of all credits will be 128 this would include the credits for the first switching unit 104a. The sum of all credits is always equal to the controlling cell internal period.

When the congestion state is L0 or L1 then all switching units 104 are granted credits such that the queue occupancy stays near zero. Since the bursty nature of the ingress traffic is unpredictable at any instance in time, any one switching unit 104 may be getting more credits than another switching unit 104. The goal being that while the system as a whole is in the L0 or L1 state, the algorithm permits large bursts from any switching unit 104. The credits are modulated in a manner such that the switching units 104 get enough credits to empty their queues. For example, it does not make sense to give credits to a switching unit 104 if it is not going to use them. The first switching unit 104a would know this from the free list feedback control word.

Upon receiving the credits, each slave switching unit 104 starts to launch cells into the upstream OC-3c link until its credits are exhausted. The slave switching unit 104 simply remains inactive until the next downstream control cell grants more credits. During the inactive state, the PHY device will insert idle cells into the OC-3c when necessary.

The slave switching unit 104 generated feedback control cell is initiated in the last switching unit 104n excluding the fields of the intermediate switching units 104i which are all

13

1's. Hardware in the intermediate switching units 104i ORs in its 32 bit feedback word, recalculates the CRC-10 and then sends the control cell to the next switching unit 104. This hardware process shall be completed within less than two cell time intervals. The software is only required to write the 16 bit feedback word at the control interval rate (i.e. for the 128 cell interval this is about 350 us).

Slave switching unit 104 generated status cells are mapped into a following standard OAM format;

Octet	Function
1 .. 5	standard ATM header
6	-4 bits OAM type -4 bits Function type coding TBD
7 .. 39	shelf_[7..0] 8 words of 32 bits contain the status for each of the 8 daisy chained shelves octets #7 to 10 are for the first subordinate shelf etc. octets #36 to 39 is for the last shelf
4 .. 46	spare
47-48	-6 bit reserved -10 bits for CRC-10

The 32 bit status word for each of the slave switching units 104 has the following format, i.e. shelf_status[7 ... 0]

Bit	Function
0 .. 9	free list size; units are soft configurable i.e. 256 bytes per unit
10 .. 11	congestive state; 0 = level 0, 1 = level 1, 2 = level 2, 3 = level 3,
12 .. 31	reserved for future use;

B. DETAILED OVERVIEW

1. Logical Architecture

Switching subsystem 100 may be either a deterministic ingress traffic multiplexer that may be distributed over a number of switching units 104 or a statistical ingress traffic multiplexer that also supports a distribution of multiplexer functions over a number of switching units 104. Switching subsystem 100 may also include advanced queuing and congestion avoidance policies. In the downstream direction, switching subsystem 100 support oversubscription with queuing and congestion avoidance policies and is permanent virtual circuit (PVC) based. Switching subsystem 100 uses a centralized shared memory ATM switch fabric. Switching subsystem 100 is preferably capable of supporting an aggregate downstream cell rate of 370,000 cells per second and an upstream burst aggregate cell rate of 222,000 cells/sec for each of nine switching units 104. The aggregate ingress rate for switching subsystem 100 is therefore 2,000,000 cells/sec. The downstream rate supports one full OC-3c.

Thus, the upstream rate supports over-subscription of the OC-3c by a factor of 5.4. The burst upstream rate can be sustained until switching subsystem 100 enters into a congestion imminent state. Cell buffers preferably provide sufficient buffer resources for queuing up to 2x1500 byte packets from the 22 ingress ports per shelf simultaneously. Thus, the architecture preferably supports 22 physical ATM ports in each shelf (i.e. two per slot).

The downstream direction also supports oversubscription. For the most part this is handled by the ATM network

14

including the CO resident ATM switch which is delivering the OC-3c to switching subsystem 100. Switching subsystem 100 supports bursts that exceed the egress bottleneck pipe capacity. In addition, two queues are provided in the downstream direction. One of these queues is intended for step function stream (i.e. UBR etc.) that may be oversubscribed. The other queue would be used for streams that require a guaranteed QoS (i.e. CBR, VBR etc.). As such, the buffers are sized to support up to 1x1500 byte packet per egress port.

The ingress architecture of switching subsystem 100 may be implemented as ingress streams or implemented in preferably 16 queues that can be assigned to up to 16 traffic classes with over-subscription. The traffic classes are organized from highest to lowest priority. Each traffic class can be further subdivided into multiple groups, however, each group preferably requires its own queue. Mixed mode scheduler operation is supported in order to provide MCR>0 for some of the lower priority queues. An example configuration, which utilizes 16 queues, could be four traffic classes where each traffic class has four groups. Switching subsystem 100 may provide a tiered congestion hierarchy and each class of service may be at a different congestion state. When switching subsystem 100 is oversubscribed, the lowest priority traffic class will enter the congestion imminent state. Switching system 100 then implements packet discard policies including early packet discard (EPD) or partial packet discard (PPD). The packet level discard algorithms operate on ATM adaptation layer five (AAL5) traffic streams. If the load offered from the remaining higher priority traffic classes remains within the OC-3 limit, then these traffic classes would not enter the EPD state.

Meanwhile, the lowest priority traffic class has its ingress rate modulated by the fairness process to the excess capacity on the upstream OC-3c. Thus, the access network will deliver nearly ideal quality of service (QoS) parameters (cell delay variation (CDV), CTD etc.) for the higher priority classes. The EPD/PPD process works in conjunction with the fairness process. In effect, the group members of the traffic class, are proportionally affected by packet level discards. Within a given traffic class multiple groups can be provisioned each with a different level of performance. This is achieved by setting up one queue for each group. The congestion policies are applied to the groups that belong to a class of service. However, provisioned parameters permit the performance to vary between the groups. For example, if two groups are provisioned for a class of service (i.e. UBR) and if the UBR class of service enters the EPD state, discards from the two ingress groups may be at different rates. The provisioned parameters for each group controls the EPD discard rate, however, in order to provide minimum throughput for each group member, a bandwidth lower limit parameter is also provided. The architecture supports per virtual circuit assignment to a traffic class and to a group within that traffic class.

Switching system 100 provides daisy chaining for a plurality of switching units 104. In the embodiments described herein, the processes apply nine switching units 104 concurrently, although other numbers of switching units 104 are possible. In other words, the switching units 104 cooperate to implement the fairness process. The daisy chaining queue (bypass) process takes priority over local queues. In effect, each switching unit 104 generates a conformant cell stream where the sum of the rates from the multiple switching units 104 equal the OC-3 rate. This results in nearly identical queuing delays for the switching units 104. Thus, there is no QoS penalty for daisy chained switching units 104.

2. Applications

The following asymmetric and symmetric bandwidths with twisted pair drops based on the ANSI (T1E1) specification, may be advantageously applied in the residential and in other environments:

Line Code	Downstream	Upstream
ADSL	6 Mbps	0.64 Mbps
HDSL	1.536 Mbps	1.536 Mbps

Switching subsystem 100 is flexible and may be characterized to support a downstream rate of 8.192 Mbps and an upstream rate of 2.048 Mbps for each residence or other drop. The specific physical interface to the home, in many cases, will have less bandwidth, and thus switching subsystem 100 is flexible to accommodate the low end rates of 128 Kbps downstream and 128 Kbps upstream. Data rates specified herein are merely exemplary, however, and other data rates may be used as well.

The following table identifies the ITU class of service definitions. Switching system 100 can support classes A, B and C. In addition, the ATM Forum has defined traffic types that map into the ITU classes, which are CBR, rtVBR, VBR, ABR and UBR. Switching system 100 may provide 16 queues that can be assigned to the traffic classes traversing through the access network. Thus, one or more queues could be used by a particular class of service. The queues are organized from highest to lowest priority.

	Class A	Class B	Class C	Class D
ATM Forum defined traffic types	CBR	rtVBR	VBR, ABR, UBR	nil
timing relation between source and destination	required		not required	
bit rate	constant	variable		
connection mode	Connection oriented			connectionless

The mapping of any of these traffic classes through switching subsystem 100 is achieved by a connection admission control (CAC) process. The CAC process should provision the channels, with their associated attributes, only when the QoS can be guaranteed to the user. Thus, the behavior of switching subsystem 100 depends on the CAC process.

3.1 Architecture Introduction

This architecture follows the spirit of GR-2842-CORE ATM Service Access Multiplexer Generic requirements. Switching subsystem 100 provides features and enhancements not required by the GR-2842-CORE. The enhancements include statistical multiplexing and virtual path/circuit switching capabilities. In a network implementing switching subsystem 100, preferably the ATM switches will use the Virtual UNI functions. In effect, the ATM switch terminates the signaling streams as defined in GR-2842-CORE and acts as a proxy Connection Admission Control (CAC) entity for switching subsystem 100. In addition, although not strictly required, the ATM switch should provide a Usage Parameter Control (UPC) (policing) function for the virtual UNI drops resident in the switching subsystem 100.

Switching subsystem 100 implements advanced functions that are generally transparent to the data path traffic. A control channel between switching units 104 permits the switching units 104 to cooperate on the various system

functions. The switching units 104 are classified as first, intermediate or last. The first, intermediate, and last shelf classes generally run similar software, but each of the classes may have its own unique processes. The architecture is capable of both VP and VC cell routing with support for up to eight or more traffic classes. The control levels provided by the MegaSLAM can be grouped into five categories although other control levels are possible. The five exemplary categories are:

- 1 instantaneous controls
 - cell routing
 - selective cell discard (EPD/PPD)
 - signaling cell mapping
 - cell statistics gathering
- 15 EFICI/CLP marking
- 2 real time controls
 - control process for cell buffer management (to declare congestion states)
 - control process for UPC of ingress streams (with virtual circuit granularity)
 - compute fairness primitives (i.e. EPD/PPD rates)
 - compute queue occupancy
- 3 hop-by-hop propagation delay controls (or segment-to-segment)
 - inter-shelf peer to peer element state signaling (i.e. for fairness algorithm)
- 4 end-to-end propagation delay controls—EFICI flow control
- 5 end-to-end round trip delay controls
 - 30 CAC I/F via NNIS
 - routing provisioning via NMS
 - 3.1.1 Over-Subscription and MegaSLAM Architecture Rationale

The downstream interface between the ATM switch 12 and switching subsystem 100 is implemented over an OC-3c pipe. This pipe can be over-subscribed by a back end ATM network associated with the ATM switch 12 and its associated Connection Admission Control process (CAC). The CAC process running in the ATM switch 12 would be able to grant bandwidth resources on this OC-3c substantially in excess of the OC-3c pipe capacity. The process would preferably rely on statistical methods to define the upper limit of its bandwidth assignment. For example, the CAC process may provision 200 user channels, each with a PCR of 1.5 Mbps, which would result in a worst case bandwidth load of 300 Mbps. However, due to statistical loading, the actual normal offered load on the OC-3c may be in the 100 Mbps range or less. In this case, no cell discards would occur in the CO resident ATM switch 12.

However, periodically, for the high demand periods during the day, an overload situation may exist for the 200 downstream user sources in this embodiment. In this case, the user sources may attempt to load the backbone ATM network to 200 Mbps or more. For the UBR traffic case, the TCP/IP protocol with its inherent rate reduction algorithms would slow down the user sources until a reasonable ratio of successful packets are getting through telecommunications network 10. In effect, the user sources in this embodiment would slow down to an aggregate rate that is approximately equal to the bottleneck rate (in this case the OC-3c pipe). Therefore, the downstream direction can be greatly over-subscribed while still delivering acceptable level of performance to each user port. If the backbone ATM network supports advanced discard policies (e.g. EPD), then the system throughput would be maximized. This is due to the one for one relationship between the discarded AAL5 packet and the TCP/IP layer packet retransmit.

Switching subsystem 100 sees the oversubscribed load (from the 200 user sources) offered on the downstream OC-3c pipe. The ATM switch 12 would fill the OC-3c, and any cells in excess of the 150 Mbps rate would be discarded by the ATM switch 12 when its buffers overflow. Fundamentally, the ATM traffic classes can be grouped into two types of streams. The predictable, traffic-shaped streams (e.g., CBR, VBR) and unpredictable, fast-rate-of-change streams (e.g., UBR, ABR). In the downstream direction, the MegaSLAM delivers the predictable traffic-shaped streams in a deterministic manner, which guarantees delivery of these cells over the bottleneck PHY. The CAC process preferably ensures that the traffic-shaped streams remain within the bandwidth bounds of the bottleneck link. Therefore, to a high degree of certainty, no cell discard events can occur through switching subsystem 100 with respect to the traffic-shaped streams. Note: Cell level discard is preferably avoided, since the discarded cells invoke packet level retransmits at packet sources, which results in an increase in the ingress rate that can quickly cause severe congestion.

The remaining unpredictable, fast-rate-of-change cell streams, which frequently are step function rate of change streams, are lower priority. Sufficient buffer capacity is preferably provided to absorb packet size bursts, but when the buffer resources are exhausted then these streams will invoke the congestion policies such as cell discard. This approach protects the traffic-shaped stream from the unpredictable behavior of the fast-rate-of change streams. For any one virtual circuit the peak cell rate (PCR) parameter for step function streams can be set at any value including values that exceed the bottleneck PHY port rate.

Ideally, there may be multiple virtual circuits, each with a PCR=PHY rate. The 64 "goodput," or actual throughput for the application, achieved over the PHY port would be a function of the traffic pattern, buffer resources, and congestion policy. A system user may empirically tune the system parameters relating to buffer size, traffic pattern, and congestion policy. The system is preferably optimized using EPD/PPD, with the system goodput preferably being in the range of 70% to 100%.

Over-subscription is desirable for the downstream circuits due to the largely client server architectures that most applications require. In the Internet case, high-bandwidth, content-rich Web pages are downloaded to the client in response to low-bandwidth upstream requests. A typical Internet application might have an optimal ratio of downstream to upstream bandwidth of about 10:1. Thus, for the client server applications, statistical multiplexing in the upstream direction would generally not be required, because the upstream link would be partially filled. For other applications, however, a ratio of downstream to upstream bandwidth may vary down to a 1:1 ratio. These applications may be web servers that are serving a web page to a remote client. In addition, if low speed symmetrical links predominate like HDSL at 384K or 768K then, over-subscription in the upstream direction becomes very beneficial. Due to the unpredictability of future applications and market demands, switching subsystem 100 preferably supports both upstream and downstream over-subscription, addressing both asymmetric or symmetric bandwidth applications. Switching subsystem 100 is intended to provide maximum flexibility. Therefore, switching subsystem 100 can evolve to address future applications.

Over-subscription in the upstream direction has been conceived to support up to 16 different types of traffic streams. The streams could be assigned to different traffic

classes or groups within a traffic class. This provides the network provider the flexibility to tariff customized services. For example two of these streams could be used for a VBR service each providing a different guaranteed minimum cell rate when the network gets congested. A distributed (daisy chained) fairness process controls the behavior of the multiple switching units 104. The process enforces the fairness and ensures that the upstream flows are compliant with the OC-3c bottleneck rate.

3.2 Top Level Functionality

This section provides a top level overview of the MegaSLAM system. All specifications set out herein are, however, merely exemplary. Other useful configurations are envisioned.

3.2.1 System Capabilities

Switching subsystem 100 may provide the following capabilities;

- downstream bandwidth of approximately 370,000 cells/sec

- upstream bandwidth of approximately 222,000 cells/sec for each shelf (uncongested state), which equates to an aggregate bandwidth of approximately 2,000,000 cells/sec for a 9 shelf system.

- 4096 downstream and upstream virtual path or circuits. This equates to 2048 full duplex communication channels

- downstream cell buffer capacity of 2K to 8K cells as stuffing options

- upstream cell buffer capacity of 2K to 8K cells as stuffing options

- upstream and downstream oversubscription

- Efficient memory management, dynamic sharing of memory resources between queues

- four-state congestion management. The states are: normal, congestion signaling, congestion avoidance with fairness, aggressive congestion avoidance.

- support for ITU traffic classes and distinct groups within these traffic classes.

3.2.2 Top Level Interfaces

Switching subsystem 100 interface to the ATM switch 12 is capable of both UNI and NNI cell formats. The mode is selected via a provisioning parameter. The daisy chain interface between switching units 104 is capable of both UNI and NNI cell formats. The mode is selected via a provisioning parameter. The switching subsystem 100 interface to the ports within each switching subsystem 100 supports UNI cell format. The ABCU card 22 interface to the Cell Bus provides a routing scheme that supports 60 slots with approximately four ports per slot or more. One code is reserved for broadcasting to the cards (i.e. OFFH) and is intended for embedded functions like software download.

In the upstream direction, the SBI interface 36 to the SBI bus on the ABCU card 22 will support either Cell granularity payload or SBI with DS-0 granularity payload (i.e., legacy TDM traffic). The ABCU card 22 will provision each upstream point-to-point TDM bus with one of these modes. In addition logic will be provided on the ABCU card 22 that maps cells into SBI granularity streams that are mapped over the time slot interchange (TSI) cable to the existing digital loop carrier system 20. For example, the DS-1 payload can be transported to a customer premises equipment (CPE) through the existing TDM infrastructure. For this example, the transport of the ATM cells is transparent to the existing digital loop carrier 20 equipment and the CPE equipment is used to terminate the ATM protocol stack.

Similarly, the ABCU card 22 will provide the capability to source downstream SBI-rate cell streams over the existing

SBI bus. Then, both the SBI upstream and downstream bus can be used to transport a T1-cell-mapped payload over an existing TDM network to remote CPE equipment, which then terminates the T1 cell compatible payload. In this case, the existing T1 line cards are reused to support communications protocols including ESF format and B8ZS line code.

Implementation of the designs may be through any combination of ASIC (Application Specific Integrated Circuits), PAL (Programmable Array Logic), PLAs (Programmable Logic Arrays), decoders, memories, non-software based processors, or other circuitry, or digital computers including microprocessors and microcomputers of any architecture, or combinations thereof. One embodiment preferably has a single upstream queue and a single, provisionable, fixed-rate scheduler that launches cells into the OC-3 trunk. In addition, the data structures leave room for future expansion.

3.2.3 Downstream Top Level Flows

The congestion buffer management policy for this direction is preferably a two state policy, where the two states are normal (uncongested) and congested.

The cell arriving from the ATM switch 12 is evaluated against the 2000-virtual-circuit database resident on the ABCU card 22 in the first switching unit 104a. If a match is found, then the cell is forwarded to the appropriate port on the current switching unit 104a. If no match is found, the cell is forwarded to the daisy chain OC-3c link. This approach reduces the cell rate on each hop in the daisy chain. Some of this free bandwidth may be used by control cells on the peer-to-peer inter-switching unit signaling channel. The interleaving of these control cells is expected to be about one control cell every 128 cells. Thus, a control cell is sent every 350 μ s. A byte-wide hardware register preferably supports provisioning of the control cell rate in the range of 32 cells to 2048 cells with 8 cell granularity.

Switching subsystem 100 expects that the scheduler in the ATM switch will queue cells on the OC-3c with reasonable time domain characteristics. Important ATM WAN network parameters are cell delay variation (CDV) and cell clumping characteristics. These parameters will limit the buffer requirements for the two ABCU card 22 resident queues for each egress link. The average rate for the downstream VC should normally be constrained by a given peak cell rate. Thus, the average downstream cell rate should not exceed the capacity of the physical medium. However, real-time cell arrival variations are preferably accommodated by FIFO queues resident on the ABCU card 22, two for each egress port. For rate adaptation purposes, the egress line cards will also provide a single FIFO buffer on each port to accommodate the inter-arrival time variations resulting from the shared downstream cell bus and the feedback state signaling over the TDM cell bus (about 8 cells). Thus, the large centralized queues are implemented on the ABCU card 22 and the smaller FIFOs on the ADSL line card 24 are tightly coupled with the ABCU card 22 to guarantee the bus level cell transfer behavior.

Cell clumping for ATM switched networks is not well understood by the industry. It is a function of switch loading and number of switches in the path of the VC. The large difference between the ingress and egress link rate maximizes the problem. For example, with two orders of magnitude difference between OC-3 ingress and T1 egress it would be possible to receive multiple cells from the OC-3 link during one T1 link cell time (about 275 μ s). The severity of this cell clumping is not well understood, but if near zero cell loss ratio (CLR) is a goal, then the buffer sizing should accommodate the worst case scenario. In addition, multiple UBR virtual circuits could be provisioned with PCR=drop

port line rate (e.g., T1). For this case, the sum of the ingress OC-3 rate can far exceed the T1 link egress rate. Thus, these classes of service require a separate queue. In effect each downstream port has a high priority and a low priority queue.

ATM switch 12 may produce on the order of +/-3 ms worth of cell clumping, which means the cells may arrive 3 ms ahead or behind the ideal cell position. This suggests that 6 ms worth of cells can arrive at nearly the same time on the OC-3c. The conforming streams will be queued into the high priority queue. The following buffer sizes are preferred in this embodiment, although other buffer sizes are possible:

Downstream Line Rate	High priority Buffer Size for 6 ms clumping	Low Priority Buffer Size for step function PCR Burst
6.0 Mbps	84 cells	64 cells
1.536 Mbps	22 cells	32 cells
256 Kbps	4 cells	32 cells

In one embodiment, buffers may be shared between ports based upon a statistical allocation, resulting in a significant memory savings.

The high priority buffer is preferably used for conformant (i.e. traffic shaped) streams. Examples would include CBR and VBR. The low priority buffer is used preferably for step function streams like UBR (or flow controlled streams like ABR). A buffer of 32 cells is sufficient for 3x500 byte packets or one 1500 byte packet (64 cells provides double the number of packets). The high priority buffers may never overflow, thus a discard policy may not be for this queue. The low priority buffers may be implemented with a dynamic buffer sharing process having, for example, a total downstream buffer size of 8000 cells. The high and low priority buffers for the ports share this pool dynamically. The maximum buffer occupancy of the high priority streams is approximately equal to the worst case cell clumping event. The normal buffer occupancy for the high priority streams would preferably be low. Thus, the bulk of the 8000 cell buffer would be available for the step function UBR streams or flow controlled ABR streams.

The discard policy in the downstream direction for the low priority buffers may be early packet discard (EPD) or partial packet discard (PPD).

The PPD process monitors the downstream low priority buffer and implements a random cell discard for a cell that is in the discard eligible state but saves the context. The PPD logic then searches for other cells that belong to the same packet and discards each of them through to the end of the packet. A number of Discard logic circuits may be shared between the virtual circuits. A centralized pool of discard logic blocks can then be allocated to perform PPD discards for a large number of egress virtual circuits. The EPD process is similar to the PPD process but it searches for a packet boundary before starting to discard the next packet. This packet boundary for AAL5 is indicated by the EOM cell.

Discarding traffic at the network egress is an undesirable network characteristic. Most networks should be engineered by the carriers such that statistical multiplexing and other procedures at the network ingress discards the necessary traffic. Due to the desire to maximize the efficiency of networks, avoiding the egress network discards may not be possible. For example, the egress bottleneck may be oversubscribed by step function streams such that the sum of the PCR exceeds the capacity of the downstream pipe. (e.g., 6 Mbps ADSL pipe shared among 10 UBR virtual circuits each with a PCR of 1 Mbps)

The scope of downstream fairness is between the active VCs going to the same drop, since this is the bottleneck that is being shared between the VC's. Therefore, each downstream drop may be in a different congestion state as a function of the load offered by the network. The memory management process may use a shared cell buffer pool. However, in order to prevent one drop from using more than its fair share of buffer resources, an upper limit will be enforced on the queue size for each of the drops.

If the downstream cell is not decoded by the ABCU local lookup procedures, then it is by default bypassed to the daisy chained OC-3 port. All ingress cells in the downstream directory are evaluated by the ABCU card 22 validation lookup procedure, and if it is a valid cell destined for one of the local ports then the per VC accounting policy may be enabled. This policy may supersede any discard procedure in order for MCR to be greater than 0. The EPD or PPD discard process is implemented before the current cell gets to a queue. Thus, for MCR to be greater than 0 for a given virtual circuit, the discarding of a given VC should not be permitted until its minimum throughput level has been reached. After this point, discards on the VC is permitted.

3.2.4 Upstream Top Level Flows

The cell arriving from each port is evaluated against the 2000 virtual circuit database resident in the switching unit 104. If a match is found, then the cell is queued for eventual forwarding on the OC-3c port on the current switching unit 104. If no match is found, the cell is discarded, but these discard events are logged. The supported number of ingress ports is preferably 22. The ingress burst cell rate generally is limited by the slotted SBI bus rate. For the 60 busses, this rate is 222,000 cells/sec. Therefore, the cell processing time is about 4.5 μ s. The rate that the cells are launched into the OC-3c is a function of the fairness process. Multiple switching units 104 share the upstream OC-3c to the ATM switch, and as such each shelf generates a conforming cell stream. The sum of the conforming cell streams, one from each shelf, when summed will be less than or equal to the OC-3c rate. Thus, daisy chained OC-3's are partially filled. Some of this free bandwidth may be used by the upstream peer-to-peer inter-switching unit signaling channel for transfer of control cells. The interleaving of these control cells is about one cell every 128 cell slots. Thus, a control cell is sent every 350 μ s. Preferably, the upstream feedback cell is only generated in response to the downstream command cell sent from the first switching unit 104a in switching subsystem 100.

The congestion buffer management policy may be a four state policy that implements an effective congestion avoidance process. Another congestion policy may be a single state policy implemented without congestion avoidance processes. Thus, when the buffer resources are exhausted the ingress cells are discarded.

The buffer management will preferably be statically (rather than dynamically) provisioned for an aggregate ingress buffer size. However, within this aggregate buffer, the ports can share this pool. The static provisioning prevents interaction between the upstream and downstream directions. The buffer management is preferably fully dynamic where the buffer resources are shared between upstream, downstream and bypass ports.

The cell arriving from the plural ports are first recovered from the TDM bus by a double buffered cell FIFO. As soon as a complete cell is recovered from the TDM bus, a cell available state is indicated by the logic. A round robin scanner then queues the ingress TDM-recovered cells for VP_descriptor processing. This logic checks that the VC is

valid, translates the ATM header, adds one or more additional control fields, and forwards the cell to one of the queues.

ABC card 22 may use a single upstream queue and a single, provisionable, fixed-rate scheduler that launches cells into the upstream OC-3. The fixed-rate scheduler for each switching unit 104 should be provisioned to consume a subset of the upstream OC-3, which represents the particular switching unit 104 portion of the total upstream bandwidth. For example, if the total upstream bandwidth for a four switching unit 104 is limited to 50%, then the fixed-rate scheduler in each switching unit 104 should be provisioned for 12.5%. Bursts, for a given switching unit 104 in excess of the 12.5% would thus be absorbed by the single queue in each switching unit 104. However, the burst duration should be small due to the QoS impact in the composite single queue. This approach enforces an open loop fairness scheme where a limited amount of oversubscription can be tolerated.

It is also possible to provision the fixed-rate schedulers on the switching unit 104 to the same value (for example 50%) of the OC-3. For this mode, the switching units 104 still share the 50% bandwidth, although, any one switching unit 104 may burst up to 50%. This may be achieved for example by a counter mechanism where each switching unit 104 is responsible to monitor the upstream traffic from the downstream switching unit 104. Any one switching unit 104 can only fill the upstream pipe to the maximum provisioned. The behavior in this case would be less fair if the available upstream bandwidth (in this example 50%) is oversubscribed. A limited amount of oversubscription would tend to favor the last switching unit 104n in the daisy chain. If, however, it is never oversubscribed, then the single queue in the upstream direction is always virtually empty. In general, the last shelf empties its queue by injecting its cells into the upstream slots. Unused slots would be filled with idle cells. The next switching unit 104 in the daisy chain empties its queue by injecting its cells starting after the last occupied cell slot until its queue is empty (these are the idle cell slots). This process continues until the switching units 104 are done.

The delay of the data path cells is not a function of the number of daisy chain hops. The delay is primarily a function of the conformant stream generation logic in each switching unit 104. This delay is incurred once per data path (i.e. virtual circuit). The resultant delay is therefore nearly identical regardless of switching unit 104 location in the daisy chain configuration.

An example application of the use 16 assignable queues for the ingress streams is shown in the following table.

Ingress_queue_0 to 7	spare queues
Ingress_queue_8	UBR with fair performance
Ingress_queue_9	UBR with good performance
Ingress_queue_10	VBR with MCR = 64 Kbps
Ingress_queue_11	VBR with MCR = 128 Kbps
Ingress_queue_12	VBR with MCR = 256 Kbps
Ingress_queue_13	VBR with guaranteed 100% throughput
Ingress_queue_14	real time VBR
Ingress_queue_15	CBR

In the above example queue 8 & 9 are used to support two different UBR groups. Both groups are always in the same congestion state. When the system is in the uncongested state (normal state) both groups operate identically. However, when oversubscribing, both UBR groups would frequently be in the congestion imminent state with the early packet discard (EPD) process active. The two groups can then be provisioned with different discard rates.

The cells are removed from the 18 queues based on the provisioned scheduler process and are forwarded to the OC-3c. The back end logic then generates a conformant OC-3c stream. FIG. 5 provides a block diagram of memory queuing fabric of switching subsystem 100.

The fairness process will generate conformant streams in each switching unit 104. Each switching unit 104 in the daisy chain treats the upstream daisy chain ingress stream as the highest priority stream and queues the cell in the Bypass_queue. The locally generated conformant stream is derived from the output side of the Ingress_queue [15 . . . 0]. A credit based process defines the number of cell slots that a given shelf may use, and the scheduler determines which queues get serviced. The logic on the ABCU card 22 generates the conformant stream by launching the permitted number of cells during the current control period. Assuming the control period is equal to 128 cell times on the OC-3c, then each shelf is permitted to launch its portion of the 128 cell budget. The credit based scheme keeps the physical OC3 pipe from becoming a bottleneck in any of the daisy 20 chained links.

The fairness process, and its associated credit based control function, for the multiple switching units 104 should be based on a control interval fast enough such that ingress cell exposure does not consume more than a small fraction such as approximately 5% of the total buffer resources. It is believed that a stable (non-oscillating) process is possible if the rate of change of the aggregate cell buffers is limited to a small number i.e. <5%. The planned aggregate cell buffer size is 8K cells. Thus, five percent exposure would be about 400 cells. If the ingress rate is worst case 1.0 us per cell then the control process should be faster than 400 us. 30

Various implementations of the candidate fairness process are possible. The implementation may be based on free list size (buffers available). In addition, a more advanced process may include a free list rate-of-change parameter. The process could also be based on individual queue occupancy. The overall goal of the process should be to provide satisfactory fairness between the multiple switching units 104. In some embodiments an error ratio of +/-5 percent may be acceptable. 40

The problem becomes more complicated when significant delay exists between the switching units 104 in the daisy chain configuration. If the fairness process control interval is 350 μ s, and the round trip delay to the switching units 104 is significant, then the control processes on the switching units 104 will be phased with respect to each other. The phasing is expected to be about 160 μ s for a 10 mile optical link. Reserving cell buffers for the maximum in-flight cell exposure expected between the phased switching units 104 in the system may help to ensure sufficient buffer space. 50

Ingress cells in the downstream directory are evaluated by the ABCU card 22 circuit validation lookup procedure, and if there is a valid cell destined for one of the local ports then the per VC accounting policy may be enabled. This policy may supersede any discard procedure in order for MCR to be greater than 0. The EPD or PPD discard process is implemented before the current cell gets to a queue. Thus, for MCR to be greater than 0 for a given virtual circuit, the discarding of a given VC should not be permitted until its minimum throughput level has been reached. After this point, discards on the VC is permitted. 60

3.3 Instantaneous Cell Controls

The instantaneous cell control procedures that are applied on a cell-by-cell basis. Examples would include decisions as a function of the ATM cell header. Also, instantaneous memory management decision fall under this category. This 65

would include taking a buffer from the free list and appending it to a queue.

3.3.1 Signaling and Virtual Path Cell Routing

The signaling VCs from each of the end users can be tunneled through switching subsystem 100 to the CO resident ATM switch 12. The tunneling approach maps the default user signaling virtual circuit (VC=5, VP=0) to the ATM switch 12 as a function of the provisioned VP_descriptor, which translates the address toward the ATM switch (the approach may be VC=5, VP=geographic slot address+port number). The value VP=0 is preferably not used. The scheme may support up to four interfaces per card or more. The CO ATM switch 12 UNI treats each of these virtual circuits (VC=5, VP=x) as signaling channels. Switching subsystem 100 does not terminate the signaling channels. The mapping function for the signaling streams is implemented by the VCI/VPI header translation logic for the supported 2000 virtual circuits. Thus, each port consumes a single virtual circuit translation resource for the signaling channel mapping to the UNI on the CO-resident ATM switch 12.

The ILMI channel from each UNI port are also tunneled through switching subsystem 100 to the CO-resident switch. The ILMI circuit (VC=16, VP=0) is remapped using the scheme identified for signaling remapping above. Thus, the CO ATM switch 12 sees VC=16, VP=X. Therefore, each port consumes a single virtual circuit translation resource for the ILMI channel mapping to the UNI on the CO resident ATM switch.

In one embodiment, the well-known VCs (VC=0 to 31) could be tunneled to the ATM switch 12, although this could impair switching subsystem 100 access to these VCs.

Within the VP address range, switching subsystem 100 is provisioned for the required number of PVCs. In the event SVC support is required, the mapping scheme described above (for the signaling channel) could also be used to provide SVC capabilities. This is referred to as VPI tunneling, and each user port is mapped to the ATM switch 12. This scheme uses the VPI address bits to uniquely identify each user. If a virtual path connection is provisioned in the VP_descriptor, then only the VPI bits are used to route the cells between each user and the ATM switch 12. The remaining VCI bits are available for SVC/PVC connections to the user end points. In this implementation, preferably the virtual path connections are unique and no virtual circuit connections will reside with the VP address range (i.e., the VP_descriptors are mutually exclusive). For the virtual path scenario, the CAC process runs in the ATM switch 12 and provisions circuits, PVC or SVC, using the VCI field. 50

The mapping function for the signaling cell routing is implemented by a hardware VC-descriptor sorted list lookup on the ABCU card 22. The ABCU card 22 resident CPU maintains a database that provisions the VC_descriptor of the ingress streams from the I/O cards and a second VP_descriptor for the egress cell stream from the ATM switch 12. This database can be provisioned in cooperation with the virtual UNI resident in the ATM switch 12. The virtual UNI in the ATM switch 12 terminates the Q.2931 signaling streams.

In addition, the interface to the ATM switch 12 port can be provisioned to support the NNI cell header format. In this case, the mapping scheme defined above is extended to support more VPs per shelf and N daisy chained shelves.

3.3.2 Data Path Cell Routing

Switching subsystem 100 preferably provides a conformant cell stream (i.e., a cell stream within characterized

bounds) for the downstream and upstream data path for each end user (UNI). Switching subsystem 100 uses the mappings in support of the virtual UNI within the ATM switch 12. The switching subsystem 100 policies and processes provide the control necessary to achieve the conformant behavior. Also, a percentage of nonconforming ingress traffic from one or more user(s) can be tolerated without affecting the QoS of conforming users. The ATM switch 12 and switching subsystem 100 are expected to cooperate via the NMS so that both entities have the access to the required database of information.

3.3.2.1 Downstream Protocol

The logic resident on the ABCU card 22 implements the cell routing function for any ingress cells from the network OC-3c, the daisy chain OC-3c, or the Upstream TDM bus stream. Virtual circuit validation is a two stage process.

The first stage logic of the virtual circuit validation process checks to see if a VP connection is provisioned for this ingress cell. Each ingress interface may be provisioned to support either UNI or NNI interfaces. The virtual path lookup is preferably a linear table where the 8/12 VP bits point to the VC-descriptor. Thus, a table with 256 byte or 4000 byte VC_descriptor entries would be used. The VP_descriptor contains the required connection information. If the virtual path lookup is successful, then the cell level processing is implemented, and the cell is forwarded to the appropriate destination. This linear lookup is fast and VP_lookup_failure indication preferably should be signaled to the next stage within a few clocks.

The virtual circuit lookup sequence is triggered by the VP_lookup_failure indication from the previous state. The virtual circuit lookup is preferably implemented in hardware by a sorted list that supports 4000 or more virtual paths. The process starts near the middle of the list and tests to see if the current 24/28 bit virtual circuit bit pattern is equal to, greater than, or less than the pattern from in the VC_descriptor entry. This hardware test is fast, preferably producing a result within 2 clock cycles. At 50 MHz, this rate permits 25 iterations of 40 ns per iteration within the 1.0 us deadline. For a VC range that is a power of 2, the number of iterations is equal to the exponent plus one (e.g., 2¹¹ supports 2K virtual circuits which requires 11+1=12 iterations). This performance may allow this architecture to be reused in future OC-12 applications while supporting 64000 virtual circuits or more.

The virtual circuit and virtual path lookup procedure preferably utilizes the same database structure named VP_descriptor. The ingress cell arriving from any port is evaluated by the VP lookup sequence first, and then a VC lookup is performed. The first successful event halts the process. Successful events invoke the header translation procedure on the permitted number of bits and the enqueueing procedure for the target queue. Any VC that falls within the reserved range (i.e. the first 32 VCs) can be passed from the main (or first) switching unit 104a to the CPU of switching subsystem 100. One approach would be to terminate these VCs in the main (or first) switching unit 104a, which could act as a proxy for the other switching units 104 in the switching subsystem 100. In addition, any inband cell that has special attributes defined in one of the control fields can cause this cell to be stripped out of the data path. Examples for this case are an ABR RM cell or an end-to-end OAM cell.

In the event the current cell is not decoded by a given switching unit 104, then it is passed via a separate FIFO to the next switching unit 104 in the daisy chain. If, however, a current cell is not decoded in the last switching unit 104n,

then this state is preferably flagged and the miss-inserted cell is passed via a separate FIFO to the CPU of switching subsystem 100.

Upon finding a valid VP or VC cell, the logic preferably writes the cell to one of the target queues. The target queue address is provided by the VCD. Each queue is built from a linked list of buffers, which are anchored by the queue descriptor. In memory, the cell consists of 52 octets, excluding the HEC octet. A buffer descriptor may be used to create the linked list for each of the queue descriptors. When a cell is eventually broadcast on the downstream bus, a routing tag is added to identify the target port. Since there is a one for one association between the queues and the ports, the scheduler can blindly generate this routing tag.

Each ADSL line card 24 preferably provides one register set for each port on the ADSL line card 24. The register may be used to determine whether or not the port needs to capture the cell on the downstream bus. The word coding scheme is set out in the control word format. A second register is provided for system specific communication (e.g., software download). A third default value may be implemented on each port card. This third default value is preferably reserved for system specific broadcast communication.

The queue structure in the ABCU card 22 supports a backplane flow control scheme between the FIFOs on the ABCU card 22 and the ADSL line cards 24. Preferably, the FIFO size on the ADSL line cards 24 is minimized such that these control cards can be implemented in ASIC. Most Utopia devices provide a two or a four cell FIFO; thus, the deadline for service in the preferred embodiment is one cell time for the PHY devices.

The feedback scheme from the ADSL line cards 24 is implemented over the upstream point to point slotted TDM cell bus. The worst-case cell rate in the downstream direction is a function of the rate adaptation circuit. The rate of the feedback scheme determines the optimal size of the local cell buffer. The design goal is to minimize the local cell buffer size, preferably keeping it within 4 or 8 cells without compromising performance).

3.3.2.1.1 Congestion and Discard Policy

The downstream buffer resources are preferably organized into a free list of buffers. The size of the buffers is a provisioned parameter, but during system runtime a single fixed size would be used. The size may, for example, be 64, 128, 256 or 512 bytes. The cells are mapped into the buffers, for example, as 52 bytes. The free list of buffers has three trigger levels plus one normal level, which are set out in the table below.

Congestion level	Level Intent	Functions
Level zero (L0)	Normal state	All cell streams are queued and forwarded to target ports
Level one (L1)	Trigger status signaling	CLP marketing EFPI marking Future BAR procedures or credit based flow control procedures discards policies on a selective basis
Level two (L2)	Congestion imminent	- early packet discard - partial packet discard - fairness process with per class or per group granularity Future enhancements per class or per group differentiated procedures
Level one (L1)	Congestion State	EFPI marking Discards policies on a - selective basis

-continued

Congestion level	Level Intent	Functions
		- early packet discard - partial pa granularity packet discard - with per class or per group granularity
Level three (L3)	Congestion	- discard CLP marked cells aggressive discard policies - cell level discards per group or class granularity. Goal: at all cost protect the highest priority QoS guaranteed streams.

If level zero (L0) is active, then the ingress cells are enqueued in the queues as a function of the VC-descriptor queue parameter. The queues are serviced by a scheduler, which may provide service policies. In the event any VC or VP connection exceeds its provisioned rate, then CLP marks the cell. The per connection accounting processing function is done in conjunction with the VC/VP lookup for the current cell. If level one is triggered, then EFCI marking is implemented on the programmed number of virtual circuits destined to the low priority queues. In addition, if level one (L1) is triggered, then the EPD/PPD procedure operates on an ingress cells for the low priority queue. The total number of EPD/PPD circuits implemented are shared among the egress ports. Each egress cell is associated with a VP_descriptor and the target queue control function is defined in the Q_descriptor (QD).

The aggregate of the upstream VCI/VPI are evaluated against the active EPD logic elements that are shared with the ports. These EPD logic elements store the context of the in-progress packet discards. If there is a match, then the EPD or PPD procedure is implemented. In other words, the cell is not queued in the target low priority queue. A pipelined implementation is preferably used wherein the VC-descriptor lookup occurs and a primitive is appended to identify the target queue and source port. The next state in the pipeline evaluates the cell to match it for a discard VCI/VPI in progress for the given port. TBD packets destined for one of the queues thus can be in the discard mode until the end of message (EOM) marker state. The EOM cell itself may or may not be discarded. The action of writing the EPD_cnt[] register sets a go command flag. The initialization of the EPD_cnt[] registers is implemented by a write cycle to the register.

While the system may be at one congestion state, the drop PHY port queue may be at a different state. Therefore a second level of congestion, namely the port congestion, exists in the downstream direction. The free list is fairly managed in a manner that gives the two downstream queues access to system resources during the uncongested system state. Each queue, however, is preferably limited in the buffer resources that it can consume. In the event the queue runs out of buffer resources, then the queue preferably defaults to cell level discard at the queue ingress.

Switching subsystem 100 supports both VP and VC connections. The EPD/PPD discard strategy is preferably used when the streams are encoded using AAL5 or a similar scheme. Otherwise, the system preferably performs cell level discards only when that stream exceeds its permitted rate. The VP connections consist of unknown VCs and provide a statistically multiplexed traffic stream that remains within some bandwidth limit. Thus, it is reasonable to discard cells if the VP stream exceeds these limits. In the VC

case, on a per VC basis, the system may be provisioned with the AAL attribute when the PVC connection is established. Therefore, only the AAL5 (or similar) encoded streams are candidates for the EPD and PPD discard strategy. Other VC streams are preferably managed with cell level discards.

3.3.2.1.2 Downstream Traffic Shaping

FIG. 6 shows a block diagram of the fabric controls for ABCU card 22. The ABCU card 22 preferably provides a programmable timer based rate adaptation circuit to traffic-shape the flows to the ADSL line cards 22. The purpose of the circuit is to rate adapt the switch fabric cell rate to the port cell rate. A set of registers is provided on the ABCU card 22 to provision the scheduler rate for each of, for example, 60 ADSL line cards 24 (x2 for the number of ports per card). Two bits are preferably used to control the rate adaptation circuit for each port. The two bits may be encoded as follows;

bit_value[1..0]	traffic shaper rate (Mbps)	Cell Repetition rate (us)
3	16.384	26
2	8.192	52
1	4.096	104
0	2.048	208

Slower rates are generally not needed because the feedback scheme over the cell slot mapped TDM bus is preferably expected to be fast enough such that at most two cells get queued for rates below 2.048 Mbps. This scheme in effect reduces the burst cell rate to each ADSL line card 24. Thus, it will be possible to minimize the size of the FIFOs on the ADSL line cards 24 and at the same time guarantee full throughput without entering the FIFO overflow or underflow state.

ADSL line cards 24 with different PHY capabilities may be used and depending on the ADSL line card's 24 throughput and FIFO resources, software may be used to provision the cell rate for each PHY drop. For example, an ADSL line card 24 that has a single ADSL interface which runs at 6 Mbps downstream would use the 8.192 Mbps cell rate. An ADSL line card 24 that has two HDSL interfaces running at 1.544 Mbps would use the two separate traffic shaped streams running at 2.048 Mbps rate.

The timers for the rate adaptation circuit are preferably designed such that they are not all expiring at the same time. In other words, multiple reset (or parallel load) phases may be implemented, possibly four or eight phases. Timers may be split between these phases.

The rate adaptation circuit signals the scheduler for each port with the state of the port buffer on each module. The scheduler for each port can then provide a cell to the port as a function of its provisioned criteria. If a cell is not delivered to the PHY port before the pipeline starves, then the ABCU card 22 TC layer function will insert an idle cell on the port. This is normal behavior when, for example, a portion of the bandwidth of the port is being utilized.

In one embodiment, the downstream 150 Mbps broadcast cell bus may be queued up to 22 cells simultaneously for the cell bus. Thus, the last cell would observe a 333 us delay and this may underflow the small FIFOs on the ADSL line cards 24. The queuing system preferably self-corrects in this condition; however, the cell bus is preferably faster than OC-3 and should be about 450,000 cells/sec. This should provide sufficient capacity above the 370,000 cell/sec OC-3c rate. Modeling can be done to ensure that the shared cell bus achieves the time domain characteristics necessary to maintain 100% port efficiency.

3.3.2.1.2.1 Scheduler

In an embodiment, the two queues for each of the up to 120 ports are controlled by a basic scheduler. The scheduling method is preferably selectable for two modes. Mode 1 is a simple priority, where the high priority queue always gets serviced first if a cell exists in this queue. Mode 2 is a modified simple priority, where the high priority queue normally gets serviced, first, however, the scheduler can periodically force that a cell gets serviced from the low priority queue. The rate is based on a timer which resets when one cell gets removed from the low priority queue and when it expires then the low the priority queue is permitted to send one cell downstream. Preferably in this embodiment, the MCR is greater than 0 for the aggregate streams in the low priority queue. The rate scheduler granularity is preferably $N \times 8$ Kbps. The scope of the rate control function applies to the LI congestion state. Each scheduler counts the number of cells sent to its port. The software may read this register and the CPU read cycle will clear the register to zero. The data may be used by the discard PPD/EPD engine to evaluate whether or not the queue should be discard eligible.

3.3.2.1.3 Encapsulated Cell Format

The ingress cells arriving from any port on the ABCU card 22 are preferably converted to a 56 byte format shown in the following table. This format only has meaning while the cell is in the pipeline and is being evaluated for a routing decision. After being evaluated, the cell is then written to memory and the format within the memory may be different. The VP and VC lookup uses that format to evaluate the current ingress cell. The HEC code has been stripped out of the stream. It is the responsibility of the PHY layers on each interface card to recalculate and insert the HEC field at every egress port.

Address (Hex)	Description
00 - 03	Control word
04 - 07	ATM header
08 - 37	ATM 48 byte payload

This encapsulated cell format is generally used on the ABCU card 22 for cell flows. The upstream TDM flow, which contains a basic routing tag identifying the source port, is converted to this format. The slot number is not encoded since the dedicated point to point SBI bus is sufficient to define the source slot. The downstream shared bus uses this format. Ingress cells from the two OC-3 ports are also converted to this format.

The VP and VC lookup circuit(s) generally use between 8 and 28 bits from the ATM header and 8 bits from the control word when evaluating the current cell. In this embodiment, the VP lookup has a maximum of 20 bits and the VC lookup has a maximum of 32 bits. The 32 bits are sufficient since the ports are restricted to UNI capability. (i.e. 24+8)

3.3.2.1.3.1 Control Word Format

When receiving cells from the many drops, uniqueness is preferably guaranteed for the streams. As such, the cell assembler logic that is recovering the cell from the SBI bus provides an overhead byte (or multiple bytes) that provides the source port and slot number. The VC and VP lookup uses this information to evaluate the current ingress cell. An example conflict is the signaling VC=5 from the ports; each one of these VCs will be remapped to a unique VCI/VPI value. These cells are then forwarded to the OC-3 toward the CO resident ATM switch 12. This switch can then uniquely

identify the signaling channel from each of the ports of switching subsystem 100. An exemplary set-up using a single overhead byte is set out in the table below.

Bit	Description
0-7	source_port []; 0-119 is shelf local port address 120-239 - reserved for shelf ports 240 - CPU address 241 - 247 - spare 248 - 250 - reserved for trunk ports 251 - trunk port (upstream daisy chain, OC-3) 252 - 254 - reserved for bypass ports 255 - bypass port (downstream daisy chain, OC-3)
8-31	spare

3.3.2.1.3.2 Memory Subsystem

The memory subsystem may, by way of example, be implemented in a 32 bit or 64 bit wide memory subsystem. The encapsulated cell format may be selected to easily map into either (or another) memory width. A 64-bit-wide single memory subsystem is preferred. For the 64-bit-wide scheme, one 64 bit control word and 6x64 bit payload words can be mapped into memory. In a cache implementation, this approach may fit into one or two standard cache line(s) depending on the line size of the cache microprocessor. Therefore this approach advantageously utilizes the memory access efficiency. Preferably, the upstream and downstream flows are unified such that future local switching between the flows is possible. For example the ATM layer could crossconnect two ports on the ABCU card 22.

In some applications, the downstream and upstream flows may be kept separate. In such an application, the ATM switch 12 behind switching subsystem 100 would preferably perform the aforementioned switching task. The preferred embodiment, however, is to use switching subsystem 100 as a switching platform wherein switching subsystem 100 behaves as an ATM switch.

3.3.2.1.4 TDM network with ATM Transport

In one embodiment, the interface to the TSI cable can source up to eight T1, ATM-formatted streams. These streams may be transported over the digital loop carrier 20 TDM infrastructure to new CPE equipment that terminates the ATM protocol. The TDM network and any cross-connects in the path generally comply with these rules:

- 1—T1's are in clear channel, i.e. 8 bits in every DS-0 available.
- 2—drop T1's are in ESF format with B8ZS line code.
- 3—cross-connects are implemented such that the differential delay for the 24 DS-0's is the same.

With this approach, the TDM network can be provisioned to transport ATM. If, however, the legacy T1 card is resident in switching subsystem 100, then the digital loop carrier 20 TSI TDM switch cross-connects the 24 DS-0's and routes them back to the switching subsystem 100 resident T1 card. In this mode of operation, the SBI bus in switching subsystem 100 operates in a TDM framed mode.

In one embodiment, the 8 T1 ATM interface circuits on the ABCU card 22 generate the ATM compliant payload for the 24 DS-0 channels without the framing bit. The ATM interface circuits map cells into the TDM frame structure, HEC CRC generation, and idle cell insertion. They may also implement some ATM layer OAM policies, such as the 1.610 and AF-xxxx policies. This may include loopbacks, alarm signaling etc. The ATM HEC framer will comply with 1.432 policy.

3.3.2.2 Upstream Protocol

Preferably, the ADSL line cards 24 in the upstream direction receive ATM cells from the PHY layer device and queue two cells for mapping over the TDM bus. One cell is assembled in memory while the other is sent over the TDM bus to the ABCU card 22. The TDM bus in this embodiment runs slightly faster than T1 rates, thus, it will take about 240 μ s to transfer one cell over the TDM bus. The extra overhead, if sufficient, can be used for circuit emulation service (CES) encapsulation of a T1 stream. Once a cell is available in its entirety, then the cell is placed in the OC-3c TDM Cell Fifo on a first come first serve basis.

The ABCU card 22 will receive up to, for example, 60 concurrent cells over the TDM slotted bus. An ID tag is also transferred over the TDM bus to indicate which port the cell came from. This ID tag is also used when more than one port is implemented on an ADSL line card 24. After receiving a complete cell from the TDM slotted bus, then the next logic stage validates the cell and provides any translation before the cell is forwarded to one of the 16 queues for eventual relaying on the OC-3c link.

The FIFO logic on the ABCU card 22 that buffers the 60 distinct cell streams also shares a common FIFO that terminates on the local CPU bus. This FIFO is used to queue OAM cells, signaling cells, and other cells to be terminated by the local CPU. Encoded with the exemplary 52 byte cell is additional overhead, including, for example, the port number the cell was received from. The VCD lookup process is also required to scan the 60 TDM cell assembly buffers for valid overhead cells that require removal from the cell stream. The queue intended to pass these cells to the CPU should be large (e.g., 32 cells). Even though the control/signaling cell rate is slow, it is possible that multiple control cells arrive simultaneously from many ports.

Similar to the downstream direction, the data path logic in the upstream protocol implements a two stage routing decision. First the VP routing stage is invoked, followed by the VC routing function in the event the VP stage failed. In the event non-provisioned cells are contained within any upstream path, they can be forwarded to the local CPU via a separate queue. The routing function on the ADSL line cards 24 may be encoded as a single byte upstream control field which may be appended, for example, to the 52 byte ATM cell. The HEC code is preferably not transported over the TDM bus. The upstream control bits may, for example, be mapped as follows;

Upstream control Byte (bits)	Description
6..7	spare
2..5	fifo_status[] - one bit for each channel card FIFO, when bit = 1 then room for a cell when bit = 0 then no room. bit 2 for channel 1, bit 3 for channel 2, bit 4 for channel 3, bit 5 for channel 4,
0..1	port_addr[] (i.e. max. 4 PHY I/Fee Statements per cards)

The cell assembly unit on the ABCU card 22 for the upstream paths will append the geographic address field and other information to conform with the encapsulated cell format. The ADSL line card itself generates the port address field. As stated, the VP and VC routing decision for the data path may be made as a function of the relevant VCI/VPI bits from the cell header. However, the header alone does not normally ensure the cell came from a unique port. Thus, the

geographic address and the port ID information is used to uniquely identify the source of the cell. The VCI/VPI field in the cell header does not guarantee that each UNI will use different values (e.g., ports might use the same signaling channel VPI/VCI). These control or signaling cells may be stripped out of the stream and presented to the CPU.

The queue structure in the ABCU card 22, which assembles the cells for the streams, supports a backplane rate adaptation scheme between the FIFOs on the ADSL line card 24 and the ABCU card 22. The ADSL line card 24 will inject data cells onto the TDM slotted bus, but when its FIFOs are empty then idle cells will be sent. The ABCU card 22 performs a proprietary scheme to ensure cell delineation over the TDM bus. This scheme will discard any idle cells that are mapped onto the TDM bus for rate adaptation purposes. The implementation goal on the ADSL line card 24 is to utilize a small buffer for the cells and to optimize throughput over the TDM slotted bus. Preferably, these functions will be implemented in an ASIC on the ADSL line cards 24, although other hardware, software, and firmware implementations are possible. Most Utopia devices provide a two or a four cell FIFO. Thus, the PHYs should preferably be serviced within one cell time.

3.3.2.2.1 Congestion and Discard Policy

The upstream buffer resources are organized into a free list of buffers. The size of the buffer is a provisioned parameter, but during system run time one fixed size may be used, which is preferably 64 byte aligned. The size may, for example, be 64, 128, 256 or 512 bytes which equates to 1, 2, 3 or 4 cells. The cells are mapped into the buffers as 52 bytes. The system level congestion state is primarily a fraction of the free list of buffer. The free list of buffers preferably has three trigger levels plus one normal level, according to the below table.

Congestion level	Level Intent	Functions
Level zero (L0)	Normal state	All cell streams are queued and forwarded to target ports CLP marking - f(x) of VC_accounting
Level one (L1)	Trigger status signaling	EFCI marking ABR procedures or credit based flow control procedures
Level two (L2)	Congestion Imminent	discards policies on a selective basis - early packet discard - partial packet discard - fairness - process with per class or per group granularity
Level three (L3)	Congestion	- discard CLP marked cells aggressive discard policies - cell level discards per group or class granularity. Goal: protect the highest priority Qos guaranteed streams.

If no levels are triggered (i.e., level zero) than ingress cells are enqueued in the 16 queues as a function of the VP_descriptor queue parameter. The 16 queues are serviced as a function of the scheduler process. The cells are then mapped into the OC-3 PHY layer (consult the conformant stream generation section). If the cell stream exceeds its VC accounting limit, then the cell may be CLP marked. If level one is triggered, then EFCI marking is implemented on the programmed number of cell streams destined to some of the queues. If the VC or VP exceeds its VC accounting limit,

then CLP marking may be implemented. If level two is also triggered, then level one procedures remain in effect. This is possible because packet level discard will occur before the cells are queued into the respective queue. The EPD procedure operates on ingress cells with port granularity. The total number of EPD circuits implemented are shared among the ingress ports. Each ingress cell is associated with a VP_descriptor and the target queue is associated with the Q_descriptor. The aggregate of upstream VCI/VPI are evaluated against the active EPD logic elements that are shared with the ports. These EPD logic elements store the context of the in progress packet discards. If there is a match, then the EPD or PPD procedure is implemented by the hardware. In other words, the cell is not queued in one of the queues (preferred of queues=16). A pipelined implementation may be used wherein the VC-descriptor lookup occurs and a primitive is appended to identify the target queue and source port. The next state in the pipeline evaluates the cell to match it for a discard VCI/VPI in progress for the given port. This means TBD packets destined for one of the queues can be in the discard mode until the end of message (EOM) marker state. The EOM cell can be provisioned or discarded. The action of writing the EPD-ctrl[] register sets a go command flag. The initialization of the EPD-ctrl[] registers is implemented by a write cycle to the register.

While the system may be at one congestion state, each of the upstream queues of the OC3 PHY port may be at a different congestion state. Therefore, a second level of congestion exists in the upstream direction, namely the OC-3 port congestion. The free list preferably is fairly managed in a manner that gives active queues access to system resources during the L1 and L2 system congestion state. However, each queue will have a limit on the buffer resources that it can consume. In the event the queue runs out of buffer resources, then the queue will default to cell level discard at its ingress.

Switching subsystem 100 supports both VP and VC connections. The EPD/PPD discard strategy is preferably used when the streams are encoded using AAL5 or a similar scheme. Otherwise, the system preferably performs cell level discards only when that stream exceeds its permitted rate. The VP connections consists of unknown VCs and provide a statistically multiplexed traffic stream that remains within some bandwidth limit. Thus, it is reasonable to discard cells if the VP stream exceeds these limits. In the VC case, on a per VC basis, the system may be provisioned with the AALx attribute when the PVC connection is established. Therefore, only the AAL5 (or similar) encoded streams are candidates for the EPD and PPD discard strategy. Other VC streams are preferably managed with cell level discards.

3.3.2.2.1.1 Congestion Control State Machine

The state machine behavior for the four congestion levels is:

```

IF L0 then
  No congestion policies implemented end;
end;
IF L1 then
  EFCI mark egress cells going to queues that are
  programmed with EFCI enable in the VP_descriptor.
  CLP mark egress cells going to queues that are
  programmed with CLP enable in the VC_descriptor
end;
IF L2 then
  EFCI mark egress cells going to queues that are
  programmed with EFCI enable in the VC_descriptor.
  CLP mark egress cells going to queues that are
  programmed with CLP enable in the VC_descriptor

```

-continued

```

If ingress cell is CLP marked then discard cell.
Else If ingress cell is a current EPD or PPD candidate
  then discard Else queue cell end;
5 end;
IF L3 then
  EFCI mark egress cells going to queues that are
  programmed with EFCI enable in the VC_descriptor.
  CLP mark egress cells going to queues that are
  programmed with CLP enable to the VP_descriptor
10 If ingress cell is CLP marked then discard cell.
  Else If ingress cell is step function type then
  discard Else queue cell
  end
end;

```

3.3.2.2.2 EPD state machine

EPD state machines preferably operate in parallel. Only one of these EPD state machines will find a match. Software should never program two EPD state machines for the same VC or VP. For ingress streams, only one state machine can be assigned to any one ingress TDM slotted cell stream. This helps to ensure that when the state machine on its own initiative finds a EPD candidate that no contention problem exists with another EPD state machine.

For ingress cells from the OC-3c the EPD/PPD state machine can be assigned to any one of the (preferably 22) sets of egress queues.

```

Do for Ingress Cell
  Do Case - Go command
  Case Search
  If last cell = COM and current cell = EOM then
    declare start_packet
    reset timer
  else
    declare ***
  end
  Case Discard
  If current cell = match parameters
    then
    discard cell
    increment cell counter
    reset timer
    If current cell is EOM then declare end-
    Packet end
  end
  End Case;
  If timer expired halt and report to CPU
  If end_packet then report status word to CPU
45 end;

```

3.3.2.2.3 Conformant Stream Generation

The upstream queues are serviced by a controller that launches a predetermined number of cells during the current control period. The upstream controller for the outbound OC3c services the upstream queues using a priority algorithm. Each queue is read until empty before advancing to the next queue. The controller blindly launches cells from the bypass_queue, and the CPU_queue since it is assumed that these streams are already conformant and have been previously scheduled by another shelf. The CPU cells are important for real time controls but are of little importance from a system load point of view. The cells from these two queues are not counted by the controller. The controller is granted a fixed number of credits for the local ingress_queue[7...0] for the current control period. As it services these queues, the credit counter is decremented until it reaches zero. At this point, the controller stops and waits for the next control period before launching any more cells. Due to boundary conditions, the controller may not reach zero before the end of the control period. The controller, when

re-initialized for the next control period, remembers the remainder from the previous period. The controller during the current period may first exhaust the counter from the previous period before decrementing the counter for the current period.

The boundary conditions impact the accuracy of the fairness process. It is expected that the delay of remote daisy chained switching units 104 may cause short term bursts from these switching units that appear to be in excess of the remote shelf credits.

Schedulers count the number of cells sent to the port. The software will read this register and the CPU read cycle will clear the register to zero. The data may be used by the discard PPD/EPD engine to evaluate whether or not the queue should be discard-eligible.

The single data path queue, the bypass queue, and CPU queue are serviced by the scheduler. The scheduler uses a simple priority process where the CPU queue gets the highest priority, the bypass queue gets the second highest priority, and the single data path queue gets the lowest priority.

3.3.2.2.3.2 Release Two Scheduler

For a multiple data path queue configuration, the data path queues plus the bypass and CPU queue are serviced by the scheduler. The scheduler may be selectable for modes which may include, for example, a first mode having a simple priority, where the highest priority queues are serviced first if a cell exists in this queue. In this mode, low priority queues may not get serviced if the higher priority stream consumes the bandwidth resources.

A second mode may be a mixed mode, where simple priority is used for N of the highest priority queues. And after these N queues are empty, round robin select for the remaining queues.

A third mode may be a mixed mode, where simple priority is used for N of the highest priority queues, but a timer interrupt for any of the lower priority queues may force that these queues get a turn. The rate scheduler is based on a timer, which resets when one cell gets removed from the low priority queue. If the timer expires, then a low priority queue is permitted to send one cell downstream. This scheme helps ensure that $MCR > 0$ for the aggregate streams in the low priority queue. The rate scheduler granularity is $N \times 32$ Kbps. The scope of the rate control function applies to the L2 congestion state. At the D congestion state, the scheduler can be disabled or can remain active.

3.3.2.2.4 Upstream Channel Card Buffering

The upstream ADSL line card 24 buffers are preferably designed to minimize delay. This is especially important for the low rate upstream rates (e.g., 128 Kbps). Thus, buffering 4 or 8 cells will preferably not be used.

A preferred approach is to buffer one cell and to start the transfer over the TDM bus at the next cell slot opportunity. A standard Utopia interface is preferably not used if it results in 2 or more cell queuing delays.

In one embodiment, the transfer of a cell over the TDM slotted bus is started before the whole cell has arrived in the local buffer. This can be thought of as a pipeline.

In applications where very low rate ingress streams occur, it may be desirable to use octet level control rather than cell level controls to minimize delay parameters. This choice will affect the preferred SBI bus cell transfer protocol.

3.3.2.2.5 TDM network with ATM Transport

The interface to the TSI cable can preferably sink up to eight or more T1, ATM formatted streams. These streams may be transported over the digital loop carrier TDM infrastructure to the switching subsystem 100 that terminates

the ATM protocol. The TDM network and any cross-connects in the path preferably comply with the following rules:

- 1—T1s are clear channel, i.e., 8 bits in every DSO available.
- 2—drop T1s are ESF format with B8ZS line code.
- 3—cross-connects effect the same differential delay for all of the 24 DSO.

With this approach, the TDM network can be provisioned to transport ATM. If, however, the legacy T1 card is resident in switching subsystem 100, then the TDM payload is preferably first routed to the digital loop carrier TSI TDM switch. This TSI switch cross-connects the 24 DS-0's and routes them back to the ABCU card 22. This mode of operation requires that the SBI bus in MegaSLAM operates in a TDM framed mode.

The 8 T1 ATM interface circuits on the ABCU card 22 terminate the ATM-compliant payload for the 24 DS-0 channels, not including the T1 framing bit. The main functions are framing on ATM cells, checking HEC, and idle cell extraction. It may also be necessary to implement some ATM layer OAM policies. (see 1.610 and AF-xxxx) This may include loopback detection and alarm signaling detection. The ATM HEC framer will comply with 1.432.

3.3.2.2.6 Usage Parameter Control

Switching subsystem 100 has per VC accounting to optimize throughput during the congestion imminent state. In addition, switching subsystem 100 will provide the following policing process selectable on a per VC or VP basis:

- GCRA—the dual leaky bucket process for VBR
- peak cell rate monitor for UBR
- ABR compliant rate monitor
- fixed rate monitor for CBR

*** UPC e.g., policing, GCRA, fixed rate and time varying rate also an aggregate per port rate.

3.3.3 Data Structures

The following subsections define the data structures shared by the upstream and downstream flows. These data structures provide the key primitives by which the architecture performs real time tasks that have very short deadlines. In many cases, the deadlines are less than 1.0 μ s.

The real time software (or alternatively firmware or hardware) provides various services to the data structures. The tasks are real time, but the deadlines are generally more relaxed by two orders of magnitude or more over a cell time which is 2.76 μ s for an OC-3. Deadlines in the range of 300 μ s to 1.0 ms will be normal for the software tasks.

3.3.3.1 Connection Control

In one embodiment, a unified data structure is defined for virtual circuit and virtual path connection control. This data structure is called the Virtual Circuit Descriptor (VCD). This architecture defines a two stage look up strategy where first the ingress cell is evaluated for a VP connection and then for a VC connection. Software provisions VPs and VCs mutually exclusive on a per port basis. The MegaSLAM switch fabric appends an overhead field that guarantees uniqueness, even if the address in the ATM cell header does not. Therefore ports can freely utilize any VP or VC address.

3.3.3.1.1 Virtual Circuit Controls

The virtual circuit cell routing process requires the implementation of a database. The data structure used for this routing decision is the Virtual Circuit Descriptor (VC descriptor, VCD). When the cell arrives from an ingress port its header contents are evaluated to determine if this is in fact a valid VCI/VPI, and routing information is appended to the cell such that the hardware can route the cell to the correct port.

The routing defining in VCD preferably supports any target queue on a given shelf. Therefore this approach supports fully functional switch fabric. This means any ingress cell can be sent to any queue including the local CPU. Local switching will be required in some embodiments, but in others cell routing will be limited to the upstream and downstream directions. Also, the ABCU itself may be a single shared memory subsystem that combines the upstream and downstream cell flows. Therefore it possible in some embodiments to forward any cell to any port (i.e. local switching).

Per VC accounting and per VC queuing control is preferably implemented with the VC-cell_cnt[] field in the VC descriptor. The purpose of these controls are to provide means to support MCR>0 for a given VC when the system enters the congestion imminent state (L2 for Release Two upstream or L1 for downstream). The field is incremented for each cell that is successfully enqueued. A background software task modifies the Time-stamp[] field when necessary to prevent roll over errors for each of the 2000 virtual circuits. The system time base counter increments at double the system frame rate or preferably 250 μs. This rate represents roughly a 9 cell exposure at the OC-3 rate. For this time base rate, rollover events for the 14 bit counter occur approximately every 4.1 seconds. Another field, VC_limit[], defines the number of cells that are to be enqueued per unit time interval before the given virtual circuit become eligible for discard policies. If the VC_limit[] field is programmed to zero, then the cells are eligible for discard policies. A global control bit VC_discard, when set, enables discards for a given virtual circuit. Assuming the port rate is 8 Mbps, then the 8 bit counter will overflow in 13.2 μs. This time period is sufficiently long, since the entire 8000 cell buffer can transition from the empty to full state in about 22 μs. Per VC accounting provides a means to enable discards, thus the real time control process is preferably at least an order of magnitude faster than the range of the resource the process is attempting to control. Assuming a 2000 cell buffer storage range for the congestion imminent state (L2), then the control process should run at 5.5 ms/10 or about 550 μs.

3.3.3.1.2 Virtual Path Controls

When virtual paths are provisioned, then controls are preferably implemented on these streams to prevent them from consuming switch fabric resources beyond some predefined limits. This helps ensure stability of the overall switch fabric.

One embodiment is to treat all VC streams that reside in the VP address range as one class of service. One approach would be to use four VPs per drop, one for each class of service. Another approach is to provision two VPs per drop, one containing the predictable streams and the other containing the step function streams. A single VP per drop poses some difficulty, since oversubscription of the step function streams (UBR, ABR) causes the QoS of the traffic shaped streams to be degraded.

In the two queue model embodiment per drop that was previously defined, each port is preferably restricted such that any provisioned VP is mutually exclusive to all VC's on the same port.

The virtual path circuits may be oversubscribed, however the switch fabric preferably will prevent these streams from monopolizing the buffer resources. The software may set arbitrary upper limits on each VP stream. The per VC accounting controller may be used to limit a VP stream to a maximum throughput per unit time.

As long as the VP streams remain within the throughput bounds defined by the per VC accounting, then the traffic is

preferably not discarded while the fabric is in the congestion state. If, however, the VP exceeds its rate, then cell level discards will preferably be implemented on *****these streams. Discard policies for VP's are generally cell based. Since the fabric generally does not have VC visibility, the EPD/PPD AAL5 discards are probably not useful.

3.3.3.1.3 Virtual Circuit Descriptor

An exemplary format of one word of the VC descriptors is as follows:

Bit position	Function
bit 0 . . . 5	target_queue[] downstream (16 - low priority queue, 17 - high priority queue) upstream (0 - ingress_queue[0] . . . 15 - ingress_queue[15]) CPU (30 - pass cell to CPU) bypass = 31
bit 6 . . . 12	spare - enough address space for per VC queuing
bit 13 . . . 20	target_port[];*** add TSI port address to map *** 0-119 is shelf local port address 120-239 - reserved for shelf ports 240 - primary CPU address 241 - secondary CPU address (delivery not guaranteed) 242-247 - spare 248-250 - reserved for trunk ports 251 - trunk port (upstream daisy chain, OC-3) 252-254 - reserved for bypass ports 255 - bypass ports (downstream daisy chain, OC-3)
bit 21 . . . 24	traffic_class[] 0-15 , user definable scheme.switch fabric uses these bits to select the congestion state for the up to 16 traffic classes.
bit 25	aal5 - when 1 then stream consists of AAL5 type, when 0 then unknown type.
bit 26	en_oam - enable terminating of inband OAM cell when 1
bit 27	en_clp - enable CLP marking when 1
bit 28	en_efci - enable EFCI marking when 1
bit 29	vc_mode - when 1 then VC mode, when cleared to 0 then VP mode
bit 30	nni_mode - when 1 then NNI mode, when 0 then UNI mode
bit 31	conn_valid - connection is valid when = 1, - when 0 then h/w ignores cell but bypasses trunk cells to daisy chain. Others are passed to CPU queue.

An exemplary format of another word of the VC descriptors is as follows:

Bits	Function
bit 0	en_EOM_discard, when 1 then signal EOM discard state to packet discard engines. when 0 then signal do not discard EOM cells to packet discard engines.
bit 1 . . . 3	spare
bit 4 . . . 31	hdr_value[] - the header translation value of the VPI/VCI field

An exemplary format of another word of the VC descriptors is as follows:

Bits	Function
bit 0 . . . 3	spare
bit 4 . . . 31	hdr_mask[] - header translation mask value of the VPI/VCI field 1's mask translation and forces setting of bit

An exemplary format of yet another word of the VC descriptors is as follows:

Bits	Function
bit 0 . . . 7	VC_cell_cnt[] - counts the number of cells enqueued per unit time
bit 8 . . . 15	VC_limit[] - defines the limit after which the cells become discard eligible.
bit 16 . . . 29	Time_stamp[] - defines the last time a cell was processed - h/w updates when cell processed - s/w task prevents roll over errors
bit 30	Force_discard - when 1, discards [all] cells when VC_limit is exceeded. When 0, [all] cells are forwarded to next stage.
bit 31	en_VC_discard - when set to 1, then enables discards. This VC can enter the discard eligible state. When 0, this VC is always in the discard ineligible state.

3.3.3.2 Memory Management

The ABCU provides centralized queues that are allocated buffer resources. The buffer resources preferably are fixed granularity memory blocks of provisionable size of 64, 128, 256 or 512 bytes. The cell is mapped into these blocks as, for example, a 52 byte entity. Each cell consumes a 64 byte block leaving 12 bytes unused. The overhead is 4 bytes for the header. Note the HEC octet has been removed at the TC layer on each port.

Each queue is implemented as a simple FIFO queue. The queue consists of a linked list of buffer descriptors. The buffers could be either pre-allocated or allocated when the cell arrives. The decision as to which approach to take is a function of the cell rate and the available CPU MIPS.

The memory management address range preferably supports at least a 4 Mbyte total address range. This is sufficient for up to 64K cells (ignoring data structures), providing flexibility for future enhancements. The emulation of the large number of queues will be preferably implemented with SRAM or pipelined burst mode SRAM. These devices are currently available at 32Kx32 which is 128K bytes. Ignoring data structures, one of such devices is capable of storing 2000 cells.

3.3.3.2.1 Memory Management Concept

FIG. 7 shows a block diagram of the exemplary memory management performed within AECU card 22. FIG. 8 shows a block diagram of the logical queue structure within ABCU card 22.

In a preferred embodiment, the switch fabric memory is managed by the software as linked lists. Two types of linked lists are simultaneously supported. The buffer size for each type of list is provisionable during system initialization. The names of the two lists are small_buf and large_buf. The supported buffer sizes are 1, 2, 4 or 8 cells per buffer. Due to the relatively slow rate of the bulk of the queues in the system, small_buf should normally be provisioned at 1 cell size per buffer. Large_buf is probably provisioned for either 4 or 8 cell size. The free list for both small_buf and large_buf is maintained by the software. When the hardware is finished with a buffer then, via a high performance CPU interface, hardware returns the buffer to the CPU. The CPU may elect to return the buffer to the either the same free list or the other free list. In addition the CPU may keep a small pool of buffers in reserve. Obviously the goal is to ensure that sufficient free list entries exist for both the small_buf and large_buf linked lists.

The MegaSLAM system consists of in excess of 280 queues, each of which has a queue descriptor to provide a reference data structure for each queue. Each hardware queue_descriptor is provisioned as to which free list to use. When the hardware needs a buffer, it preferably goes to the

tail of the associated free list and takes the buffer. Hardware then appends the buffer to the head of its linked list of buffer descriptors (BFD). The hardware can append a buffer using one of two approaches:

- 1—append buffer when last empty cell slot in current buffer is used.
- 2—append buffer when new cell arrives and last cell slot in current buffer is used.

Normally, for high performance ports, approach one is used. However, to conserve free list resources for low speed ports, approach two may be used.

Each Queue Descriptor (QD) has a defined memory location. The locations will be memory mapped in a linear address space such that the associated scheduler can easily evaluate its list of queues. The linear mapping is implemented to permit the scheduler to perform cache line reads for checking the status of its queues.

The basic idea is that cells are added to the head of the linked list, and buffers are added to the head when needed. Simultaneously, cells are removed from the tail by the scheduler. When a buffer is added at the head or a buffer is returned to the CPU at the tail, then the necessary pointers (QD & BFD) are updated. In addition, software may limit the length of the queue to prevent one queue from consuming excessive buffer resources; this is achieved by the queue_size[] field in the QD. When the scheduler returns a buffer to the CPU at the tail of the linked list, then a decrement pulse is generated to decrease the value of queue_size[].

The single pointer to the payload scheme defined in the BFD does not support the boundary conditions when only one buffer is attached to the QD with size>1 and simultaneously the scheduler wants to read a cell while the VCD wants to write a cell to the same buffer. Thus, in this case the scheduler preferably waits until the head of the queue advances to the next buffer.

3.3.3.2.1.1 Queue Descriptor

One queue descriptor is preferably associated with each queue on the ABCU. This data structure provides the reference point for the linked list of buffers that implements the simple FIFO queue.

The name of QD_name can be any of the queues (i.e. ingress_queue[0]_port[12] etc.). The encoding of this name may be the same as the encoding scheme used in the VCD. In excess of 280 queues may be active in every shelf. Each scheduler has its own subset of queues that it services based on the provisioned scheduler process.

QD_name[] bits, word 0	Function
0 . . . 21	queue_head_ptr[], pointer to head BFD of queue list,
22 . . . 27	queue_limit[], 0 = no limit IF queue_limit[] > queue_size[] (6 MSB) then disable buffer attach at head of queue
28	spare
29	buf_present, when 1 = yes; when 0 = no
30	buf_type, when 1 = large, when 0 = small
31	en_queue, when 1 = enabled, when 0 = disabled
QD_name[] bits, word 1	Function
0 . . . 21	queue_tail_ptr[], pointer to tail BFD of queue list,
22 . . . 31	queue_size[], in buffer granularity units

The queues are preferably simple FIFO implementations; as such, adding a cell to the queue is done at the tail, and

removing a cell from the queue is done at the head of the queue. The queue may consist of multiple buffer descriptors chained together in a linked list. Thus, each buffer descriptor provides the pointer to the next buffer descriptor in the linked list.

3.3.3.2.1.2 Buffer Descriptor Format

The buffer descriptor (BFD) is the data structure that points to a buffer. The BFDs can be linked together to form a queue.

BFD _i bit, word 0	Function
0 . . . 21	buf_ptr _i], i.e. pointer to payload
22 . . . 28	spare
29 . . . 30	buf_size _i], 00 = 1 cell 01 = 2 cells 02 = 4 cells 03 = 8 cells
31	Next_BFD, 1 = yes, 0 = no (i.e. last BFD) Note; when Next_BFD = 0 scheduler cannot access buffer

BFD _i bit, word	Function
0 . . . 21	BFD_ptr _i], pointer to next BFD (direction from tail to head)
22 . . . 31	spare

The following is an exemplary procedure for the hardware sequencer when adding a cell to a queue. Note: VCD provides queue address.

```

Do case add_function to queue;
For the given QD_address, read BFD buf_ptri ] (indirect read
f(x) of QD queue_head_ptr)
Write cell to BFD buf_ptri ] location (burst of 7x64 bit
words)
BFD buf_ptri ] = BFD buf_ptri ] + 1 cell slot
If BFD buf_ptri ] offset = buf_sizei ] then
    buffer is full
    /* add new BFD to head of list and update */
    QD queue_head_ptri ] = new BFD location
    new BFD_ptri ] = old BFD location
end
end
    
```

The following is an exemplary procedure for the hardware sequencer when removing a cell from a queue. Note: VCD provides queue address.

```

Do case remove_cell from queue;
For the given QD_address, read BFD buf_ptri ] (indirect read
f(x) of QD queue_tail_ptr)
Read cell from BFD buf_ptri ] location
BFD buf_ptri ] = BFD buf_ptri ] - 1 cell slot
If BFD buf_ptri ] = empty (f(x)LSB bits = 0) then
    /* buffer is empty */
    /* return empty buffer to CPU by
writing */
    pointer of returned BFD to FIFO
(readable by CPU)
    QD queue_tail_ptri ] = next BFD in
linked list update new BFD with
Next-BFD = 0
end
end
    
```

3.3.3.2.1.3 Cell Buffer Format

The buffers on the ABCU card 22 are preferably managed as 64 byte entities, and are aligned with the natural address

boundaries (6 low order bits are zero). Starting at the low order address, the first 4 bytes are the ATM 4 byte header. The next 48 bytes contain the ATM payload. The HEC code has been stripped out of the stream. It is the responsibility of the PHY layers on each ADSL line card 24 to recalculate and insert the HEC field at every egress port.

Address (Hex)	Description
00-03	ATM header
04-33	ATM 48 byte payload
34-3F	spare

Multiple of these cell buffers can be grouped together to provide larger buffers. For example, when a four cell buffer is constructed, then 256 bytes are utilized in a linear address space. Four 64 byte fields, within this 256 byte address field, contain one cell each as mapped by the table defined above. In this embodiment, 12 bytes are wasted for each of the 64 byte fields.

3.3.3.2.2 Queues

The following subsections define the preferred embodiment queues for each of the port types.

The access system bandwidth resources are provisioned using a user definable scheme for the active VCI/VPI channels. Switching subsystem 100 provides traffic policing and PCR limit enforcement. The ingress upstream rates are less than 2.048 Mbps. As such, the load that any one end point can inject is low. Traffic contract violations can thus be tolerated without greatly affecting the QoS of the remaining user population (a small amount of switching subsystem 100 resources will be reserved for this case). Switching subsystem 100 can be oversubscribed in both the upstream and downstream directions, however, the CAC process in the switch should be aware of the switching subsystem resources and bottlenecks when a new circuit is being provisioned.

The queue behavior is preferably simple FIFO for the upstream and downstream paths. A scheduler determines which TDM upstream and Cell bus downstream queue to service.

3.3.3.2.2.1 Drop Port Queues

Drop port queues are the preferred egress queue structure for the ports (e.g., 120 ports) supported on switching subsystem 100. The CPU queue is preferably a logical queue only. In other words, one centralized CPU queue is shared across 120 ports. The encoded routing tag is used to differentiate the ports, since the CPU-generated cell traffic is not heavy.

Queue Name	Priority 0 = lowest	Description
Egress_queue_0	0	used for unpredictable step function streams, discard this stream when congested
Egress_queue_1	1	used for traffic shaped predictable streams
CPU_queue	2	this queue is for the egress CPU cells

3.3.3.2.2.2 Bypass Port Queues

Bypass port queues are the preferred egress queue structure for the daisy chained bypass port supported on switching subsystem 100. The Bypass queue is a physical queue. The queues for this bypass port are described in the following table.

Queue Name	Priority 0 = lowest	Description
Bypass_queue_0	0	bypass unknown cells to next shelf, last shelf monitor mis-inserted cell rate
CPU_queue	1	this queue is for the egress CPU cells

3.3.3.2.2.3 Upstream Trunk Port Queues

This is the preferred OC-3 port in the upstream direction. In the first shelf of the daisy chain, this is the port to the CO resident ATM switch 12. The following two tables define the queue structures for a multiple ingress queue structure and a single ingress queue structure.

Queue Name	Priority 0 = lowest	Description
Ingress_queue_0	0	general purpose queue for ingress streams
Ingress_queue_1	1	general purpose queue for ingress streams
Ingress_queue_2	2	general purpose queue for ingress streams
Ingress_queue_3	3	general purpose queue for ingress streams
Ingress_queue_4	4	general purpose queue for ingress streams
Ingress_queue_5	5	general purpose queue for ingress streams
Ingress_queue_6	6	general purpose queue for ingress streams
Ingress_queue_7	7	general purpose queue for ingress streams
Ingress_queue_8	8	general purpose queue for ingress streams
Ingress_queue_9	9	general purpose queue for ingress streams
Ingress_queue_A	10	general purpose queue for ingress streams
Ingress_queue_B	11	general purpose queue for ingress streams
Ingress_queue_C	12	general purpose queue for ingress streams
Ingress_queue_D	13	general purpose queue for ingress streams
Ingress_queue_E	14	general purpose queue for ingress streams
Ingress_queue_F	15	general purpose queue for ingress streams
Bypass_queue	16	for the ingress daisy chain stream
CPU_queue	17	for the ingress CPU cells

Table for Release One queues;

Query Name	Priority 0 = lowest	Description
Ingress_queue_0	0	general purpose queue for ingress streams
Bypass_queue	1	for the ingress daisy chain stream
CPU_queue	2	for the ingress CPU cells

As shown, the CPU_queue gets the highest priority and the Bypass_queue gets second highest priority for both queue configurations. The CPU_queue carries a smaller number of cells; thus from the data path perspective the Bypass_queue has the highest priority.

3.3.3.2.2.4 TS1 Port Queues

This queue is only active on the main shelf, which is the first shelf in the daisy chain. The behavior of this queue is preferably identical to the per port drop queue. A total of 8

of such queues may be implemented to support up to 8 remote ATM CPEs. The TDM network provides, for example, transport and cross connect for these 8 streams.

Queue Name	Priority 0 = lowest	Description
Egress_queue_0	0	used for unpredictable step function streams, discard this stream when congested.
Egress_queue_1	1	used for traffic shaped predictable streams
CPU_queue	2	this queue is for the egress CPU cells

3.3.3.3 Registers

Preferred configurations for the registers are defined in the following subsections.

3.3.3.3.1 EPD/PPD Control Registers

In one embodiment, the EPD/PPD control registers for the centralized (TBD) number of discard logic blocks each have the following format:

EPD_cntl[]bits	Function
31..24	port_addr[]; encoded as per VCD
23..19	queue_address[]; encoded as per VCD
18..16	pkt_timeout[]; time-out for packet discard; 0 = 333 ms 1 = 100 ms 2 = 33.3 ms 3 = 10 ms 4 = 3.3 ms 5 = 1.0 ms 6 = 0.33 ms
15..14	7 = disable time-out mode[]; discard mode; 0 - PPD, 1 - EPD, 2 - cell 3 - reserved
13..10	discard_length[] defines the number of cell/packets to discard 0 = 1 packet or cells 1 = 2 packets or cells 2 = 3 packet or cells 3 = 4 packets or cells 4 to 1K = cells

When in EPD mode, the "go" command causes the hardware to search for an EOM cell from a given port that has the correct target queue primitive attached to it. Next, the hardware starts discarding COM cells through to the end of the packet. The hardware then decrements the packet discard counter and, if zero, sets the done flag. Otherwise, the hardware continues and repeats the process. The timer is enabled by the "go" command and cleared by any received cell from the given port that matches the EPD criteria programmed in the EPD_cntl[] register.

When in PPD mode, the "go" command causes the hardware to search for a COM cell from a given port that has the correct target queue primitive attached to it. The hardware discards this cell and subsequent cells through to the end of the packet as signaled by an EOM cell. The hardware then decrements the packet discard counter and, if zero, sets the done flag. Otherwise the hardware continues and repeats the process. The timer is enabled by the "go" command and cleared by any received cell from the given port that matches the PPD criteria programmed in the EPD_cntl[] register.

In one embodiment, the total number of PPD/EPD logic blocks (64 TBD) may be shared among the ingress and egress ports. As needed, their logic blocks may be assigned to a port to discard one or more packet(s).

EPD_status[] [bits]	Function
15	done - 1 when done, 0 when in progress
14	error - when 1 command failed due to time-out
13..0	cell_ctr[] - total number of cells discarded for current command

The hardware also may have, for example, an embedded 28-bit register that is not readable by the software. This register is used to store the context of the VCI/VPI that is in the discard mode.

In another embodiment, VC discard granularity may be used. This would permit discarding multiple VCs going to the same port. One approach is to use a sorted list that supports >64 concurrent discards. The list itself stores the VCs that are in the discard mode and a pointer to the register set that is assigned to this VC. Thus, if it is implemented in the VC pipeline, then a 1.0 us deadline permit a single discard engine for servicing the >64 discard events. With this approach, we may as well increase the limit to 256 concurrent discards.

3.3.3.3.2 Rate Adaptation Circuit Registers

The two bit values required for the rate control of the rate adaptation circuit for the 120 queues are mapped into eight 32 bit registers;

reg_port[7..0] bits	Associated slot
30..31	16 + 16 × reg_port[x]
28..29	15 + 16 × reg_port[x]
26..27	14 + 16 × reg_port[x]
24..25	13 + 16 × reg_port[x]
22..23	12 + 16 × reg_port[x]
20..21	11 + 16 × reg_port[x]
18..19	10 + 16 × reg_port[x]
16..17	9 + 16 × reg_port[x]
14..15	8 + 16 × reg_port[x]
12..13	7 + 16 × reg_port[x]
10..11	6 + 16 × reg_port[x]
8..9	5 + 16 × reg_port[x]
6..7	4 + 16 × reg_port[x]
4..5	3 + 16 × reg_port[x]
2..3	2 + 16 × reg_port[x]
0..1	1 + 16 × reg_port[x]

3.3.3.3.3 Other Registers

Other registers include a scheduler cell counter register, a BFD-to-free-list FIFO, and others.

3.4 Real Time Controls

The deadline for real time controls are about two or three orders of magnitude greater than the per cell deadline. These controls may be implemented by a RISC CPU on the ABCU card 22. The CPU cooperates with the peer CPUs in other switching units 104 that may exist in a daisy chained configuration.

The control loop may span a maximum distance of 30 Km or more, thus this limit is observed over the sequence of switching units 104. In this embodiment, the control loop has significance for upstream flows only.

3.4.1 Downstream Processes

In the downstream direction, the cells are fanned out to their target switching units 104 via the VC descriptor lookup in each switching unit 104. The cells are enqueued into

either a high priority or a low priority queue associated with each drop (or port). The ABCU card 22 is capable of up to 120 sets or more of these dual priority queues.

Each queue implements a real time buffer to attach to the queue from the free list. Hardware preferably will perform the buffer attach, software will preferably manage the free list including the congestion states. In the downstream direction, two levels of congestion exist—congestion caused by the drop port and congestion of the overall switching subsystem 100 due to finite shared resources for the drops. The overall congestion state is primary a function of the free list size. The per drop congestion state is a function of the allocated resources to the two queues and the characteristics of the cell streams. Naturally, more advanced procedures are possible.

The software memory management function preferably manages in excess of 280 queues. As stated, the hardware acquires buffers from one of two free lists. However, in order to prevent one queue from consuming more than its fair share of buffer resources, the software provides safeguards to prevent one queue from consuming more than its fair share of system resources. For example, if the overall system is in the normal state (uncongested), then it is probably reasonable to permit a queue to use significant buffer resources. An upper limit could still be defined, but this upper limit could be lowered as the system declares the progressively higher congestion levels. When the system is in the LI congested state, then the gactive queues should tend to get proportionally the same amount of buffer resources. (i.e. a 6 Mbps port gets three times the buffers when compared to a 2 Mbps port). A queue that is limited to an upper bound and reaches that upper bound may not necessarily cause the system to increase its congestion state. However, N of these queues in this state may cause the system congestion state to increase one level.

For a single queue configuration, the upstream is a single queue. The customers may elect to oversubscribe this upstream queue. In order to prevent significant interference between the upstream and downstream queues, preferably the downstream queues should utilize more than 50% of the buffer resources. It is preferable, when network traffic must be discarded, for the discarding to be at the network ingress, because a cell that has made it through multiple switches to almost the final port has consumed expensive network resources. In an asymmetric environment, it may be preferable to let the downstream direction consume 90% of the buffer resources. Alternatively, some carriers will use this system for symmetric application, and in this case approximately 75% of the buffer resources should preferably be used for the downstream direction.

An example congestion process could be:

System Congestion Level	Level Intent	Queue Size
Level zero (L0)	Normal state	2x to 4x the proportional queue size
Level one (L1)	Congestion	0.5x to 1x the proportional queue size (for no QoS guaranteed streams) - recovered queues are given to the QoS guaranteed streams (i.e. high priority)

When the software computes a new congestion state, it preferably informs the hardware as to this new state. This may be implemented by registers. The hardware can then use the state to make real-time, cell-level decisions. For

example, CLP marking would be done during VCD processing before the cell gets enqueued. The real-time software task that computes the congestion state should do so while the state is still relevant. In other words, if the software is too slow, the real congestion state may be different than the declared state. Thus, the impact of the false congestion state may be negative. It could in some cases cause the system to oscillate. A rough guideline for the software computation of the congestion state can be calculated using the following approach:

Assume that a delta 5% change of buffer resources is the maximum acceptable change. This number is small because carriers won't get the most out of an ATM system when its heavily loaded. An example of a heavily loaded system has 75% of its buffer resources consumed by the 280+ queues. Then, the 5% change could bring the system buffer occupancy to between 70 and 80%. If the consumed buffers are going up, then only 20% headroom remains; thus, the system should more aggressively perform packet level discards to try to free up more buffers. The previously-stated 5% goal would translate to 0.05×8000 cells = 400 cells worth of buffer resources. Since each shelf has a maximum 1.0 μ s ingress cell rate, this translates to 400 μ s worst case deadline for declaring a new congestion state. It is also reasonable to assume that some cells are leaving the queues. If two cells arrive for each cell that is exiting the system to a port, then the software deadline can be relaxed to 800 μ s.

When the downstream direction is in the L1 congestion state, then a pool of PPD/EPD discard engines may be used to control the queue occupancy. If the L1 congestion state covers a 30% buffer occupancy range (e.g., from 70 to 100% buffer occupancy), then the goal for the discard process should be to operate around the middle of this range (e.g., 85%) as long as the overload condition persists. The rate of discards is preferably a graduating scale from the 70 to 100% queue occupancy range (i.e., a discard rate increasing with buffer occupancy). However, as a function of the system load, the software will periodically adjust this graduating scale; otherwise it would tend not to remain in the middle of this range. The deadline for this adjustment is about 2 to 3 times longer than the deadline for the congestion state declaration, or about 2 ms. The controlling software drives these discard engines to fairly discard the active low priority queues in the system. The discards should be proportional to the rate that each virtual circuit is provisioned for. If, however, some VCs have guaranteed minimum throughput, then the VC accounting hardware should prevent discards for these VCs until after their minimum throughput is enqueued. The EPD/PPD discard engines can be assigned to a queue, but if the engine does not find a candidate AAL5 packet to discard, then the queue may revert to cell level discard for the ingress cells.

The software can also read a register associated with each scheduler that provides the number of cells that this scheduler has sent to its port since the last time the register was read. This is an indication of the aggregate cell rate through the queue. The controlling software can use this data to decide which queue to target for EPD/PPD discards. Some VCs or queues may offer only marginal loads to the system. If these loads are low relative to the maximum, then these queues are entitled to less aggressive discards or not to be elevated to discard status until the system gets into high end of the L1 range say 90-95%. Thus, not only could the discard rate be graduated through the range but also the discard population (i.e., candidate queues & VCs) could increase towards the high end of the range.

Some queues may reach their cell occupancy limit and, in this case, these queues would enter the cell discard mode.

The EPD/PPD engines may still be performing packet level discards, but not at a fast enough rate for these queues. Thus, if a cell level discard is invoked, then the buffer attach to the queue does not occur.

When a downstream queue reaches its occupancy limit, then preferably the cells going to the queue are discarded. In a multi-switching unit 104 configuration, each switching unit 104 may be at a different system level congestion state. The downstream direction bottleneck is the drop port. As such, each port may be at a different congestion state. Thus, the controlling software may compute the congestion state for each port or may manage the system wide traffic flows to ensure that each port gets its fair share of system buffer resources. Since each port only has two queues, the congestion state relationships can be fixed. In this embodiment, two types of congestion states exist: one for each port, and one for the system as a whole. Preferably, when the system enters a congestion state, it reduces the allocation of buffers to the lower priority queues in the system. (as shown earlier in a table).

The congestion behavior for the two queue model is:

Congestion level	High priority queue	Low priority queue
Level zero (L0)	enqueue cells	enqueue cells
Level one (L1)	enqueue cells	PPD/EPD with potential cell discards. f(x) of queue occupancy and graduated scale in L1 range.

Switching subsystem 100 supports both VP and VC connections. The EPD/PPD discard strategy is preferably used when the streams are encoded using AAL5 or a similar scheme. Otherwise, the system preferably performs cell level discards only when that stream exceeds its permitted rate. The VP connections consists of unknown VCs and provide a statistically multiplexed traffic stream that remains within some bandwidth limit. Thus, it is reasonable to discard cells if the VP stream exceeds these limits. In the VC case, on a per VC basis, the system may be provisioned with the AALx attribute when the PVC connection is established. Therefore, only the AAL5 (or similar) encoded streams are candidates for the EPD and PPD discard strategy. Other VC streams are preferably managed with cell-level discards. Therefore, the controlling software programs cell-level discards into the VCD for the streams that cannot be controlled with the EPD/PPD discard approach.

The process of mapping cells over the shared downstream cell bus may be implemented with a provisioned rate adaptation procedure. Feedback over the TDM bus providing the mechanism to keep the small FIFO on the ADSL line card 24 from overflowing or underflowing.

Preferably, each switching unit 104 on its own initiative, implements the congestion policies, thus each shelf may be at a different congestion level. If sufficient buffer resources are allocated to the downstream path, then interference generated by the upstream path consuming buffer resources can be minimal.

The slave switching units are generally required to participate in generating a feedback status cell that is sent to the master shelf. This cell contains the congestion state and the free list size for the downstream direction.

3.4.1.1. Control Cell Format

Two types of control cells exist in this embodiment: one initiated by the first switching unit 104a (control cell) and sent to the other daisy chained switching units 104; and another generated by the slave switching units 104 (status feedback cell) and terminated on the first switching unit 104a.

A master generated downstream control cell may be mapped into an exemplary OAM format as shown in the following table:

Octet	Function
1..5	standard ATM header
6	-4 bits OAM type -4 bits Function type
7..8	Control command word. - contain length of control cycle in cell times etc.
9..24	credit_cntl[7..0] 8 words of 16 bits contain the credit allowance for each of the 8 daisy chained shelves. octets #9 & 10 are for the first subordinate shelf etc. octets #23 & 24 is for the last shelf
25..46	spare
47-48	- 6 bits reserved - 10 bits for CRC-10

exemplary credit_cntl[7 . . . 0] format:

Bit	Function
0..9	number of cell granularity credits granted by master shelf
10..15	reserved for future use;

3.4.2 Upstream Processes

The first switching unit 104a runs a process that computes the congestion state as a proxy for the other switching units 104. The first switching unit 104a preferably operates on a fixed control period, which, for example, may be 128 cell time intervals on an OC-3c link, of about 350 us. During this time, the first switching unit 104a computes the credits for each slave switching unit 104. The sum of the credits will be 128, including the credits for the first switching unit 104a.

When the congestion state is L0, then the switching units 104 are granted credits such that the queue occupancy stays near zero. Since the bursty nature of the ingress traffic is unpredictable, at any instance in time any one switching unit 104 may be getting more credits than another switching unit 104. Preferably, while the system as a whole is in the L0 state, the process permits large bursts from any switching unit 104. The credits are preferably modulated in a manner such that the switching units 104 get enough credits to empty their queues. The first switching unit 104a may monitor the free list feedback control word to minimize the possibility that a switching unit 104 is given credits that it does not need and would not use.

The congestion state of a switching subsystem 100 (or a switching unit 104) may span multiple classes of service. As such, the lowest priority class of service may be in one congestion state (for example UBR at L3), while the next class of service is at a lower congestion state (for example VBR at L2). This may typically occur in the upstream direction.

Upon receiving the credits, each slave switching unit 104 starts to launch cells into the upstream OC-3c link until its credits are exhausted. The slave switching unit 104 then remains inactive until the next downstream control cell grants more credits. During the inactive state, the PHY device will insert idle cells into the OC-3c when necessary.

The slave generated control cell is initiated in the last switching unit 104n, excluding the fields of the intermediate switching units 104i, which are 1's. Hardware in the intermediate switching units 104i ORs in its 16 bit feedback

word, recalculates the CRC-10, and then sends the control cell to the next switching unit 104. This hardware process shall preferably be completed within two cell time intervals. The software preferably only writes the 16 bit feedback word at the control interval rate (e.g., for the 128 cell interval this is about 350 us).

The last switching unit 104n monitors the status of the bypass queue for entry into the status feedback cell. This data will be used by the first switching unit 104a to determine if excess cell slot grants are to be issued to the switching units 104. This may occur when switching units 104 are not using the upstream cell slots. Thus, switching subsystem 100 can take advantage of these unused cell slots.

3.4.2.1 Status Feedback Cell Format

An exemplary slave generated status feedback mapped into standard OAM format is shown in the following table.

Octet	Function
1..5	standard ATM header
6	- 4 bits OAM type - 4 bits Function type
7..22	shelf_status[7..0] 8 words of 16 bits contain the status for each of the 8 daisy chained shelves. octets #7 & 8 are for the first subordinate shelf etc. octets #21 & 22 for the last shelf
23..44	spare
45..46	Number of cells in upstream bypass queue of last Release Two shelf
47..48	- 6 bit reserved - 10 bits for CRC-10

exemplary shelf_status[7 . . . 0] format:

Bit	Function
0..9	free_list[]; units are soft configurable i.e. 4 cells per unit
10..11	cong_state[] - for lowest priority group of queues; 0 = level 0, 1 = level 1, 2 = level 2, 3 = level 3,
12..13	cong_state[] - for 2nd to lowest priority group of queues; 0 = level 0, 1 = level 1, 2 = level 2, 3 = level 3,
14..15	cong_state[] - for 3rd to lowest priority group of queues; 0 = level 0, 1 = level 1, 2 = level 2, 3 = level 3,

3.5 Hop by Hop Controls

The first switching unit 104a or last switching unit 104n in the daisy chain may be used to terminate F4 segment OAM flows.

3.6 End-to-End Propagation Delay Controls

Preferably, CPE equipment used with switching subsystems 100 will support EFCI flow control.

3.6.1 CAC Procedure

Switching subsystem 100 preferably is a PVC ATM system. Static provisioning of switching subsystem 100 be done via the operator console or via remote schemes as supported by a digital loop carrier. The network operator may gather statistics from the system and utilize this data to determine whether or not to admit a new PVC connection.

In the event SVC capabilities are available in the CO-resident ATM switch 12, then the CAC process running in that switch could provision SVC circuits that are tunneled through switching subsystem 100. The CAC process should, however, be aware of the switching subsystem 100 resources when attempting to determine how much to oversubscribe a given port. The CAC process may act on behalf of the LNI ports resident within the access network. This is sometimes called a virtual LNI interface.

The CAC function resident in the ATM switch 12 preferably implements the process utilizing a switching subsystem 100 multiplexer data base. The knowledge of the system and PHY bandwidth attributes in switching system 100 is supplied to the CAC process in order for it to determine if the QoS of the connections can be maintained. (e.g., when a new connection is being admitted).

When implementing CAC-based oversubscription, a policing function in switching subsystem 100 is probably needed to deal with the non-conforming streams. Switching subsystem 100 should (via the NMS) disconnect these sources. This procedure may take a few minutes, and in the mean time the QoS of the conforming users should not be degraded. In the event the network administrator decides on a different network policy, which may be acceptable depending on the traffic statistics, then other procedures could be implemented.

Embodiments of switching subsystem 100 may provide SVC and CAC capabilities. In one embodiment, the policing function will be included, but may be used to aid in discarding traffic non-conformant stream. The virtual circuit itself will remain active.

3.7 End-to-End Round Trip Delay Controls

As mentioned, some switching subsystem 100 embodiments are PVC-provisioned systems. Some embodiments include the Q.2931 signaling stack and the connection admission control (CAC) process for SVC automatic controls.

3.8 Statistics

Switching subsystem 100 preferably gathers the required PHY layer and ATM layer statistics for the two layers. In addition, local system specific statistics will be gathered such as statistics for the following events: queue trigger levels, queue occupancy events, cell level discard events, cell mis-inserted events, and events that relate to the accuracy of the fairness process. Switching subsystem 100 can provide ATM switching functions such as cell routing such that cell mis-inserted events will be logged by the system. The mis-inserted cells will be discarded. Switching subsystem 100 also logs physical layer events such as HEC CRC errors, OAM CRC errors, and loss of cell delineation.

Switching subsystem 100 may gather and report the statistics at periodic intervals as required by the PHY or at other intervals. An embedded statistic accumulation functions may be implemented to save the results in non-volatile memory (serial EPROM or EEPROM). This might include aggregate cell counts per unit time and queue occupancy statistics (e.g., congestion event counts and cell loss counts).

The system design provides large centralized per port egress queues and small queues for the rate adaptation function between the various interface line rates. Within generous cell clumping time domain bounds, switching subsystem 100 demultiplexing process is deterministic, therefore cells are extremely unlikely to be lost as a result of this process. If, however, this event occurs, it will be logged. In the upstream direction, congestion trigger levels may be logged by the system. A history file preferably will reside within the available non-volatile memory.

3.9 CPU cell handler

The CPU software/hardware interface can provide the ability to inject and remove cells from any link. The hardware provides the primitives to detect, for example, eight virtual circuit addresses for the cell receive function. This can be implemented with 32 bit registers and a mask function for each of the 8 addresses. This will permit unique or linear range VCI/VPI address detection or Payload Type (PT) detection. Some well known cell VCI/VPI values are:

VC address is 1 (for UNI I/F) meta-signaling;

VC address is 3 (for UNI I/F) for segment F4 OAM cell flows (segment VP flow);

VC address is 4 (for UNI I/F) for segment F4 OAM cell flows (end to end VP flow) Not needed in MegaSLAM but is required in CPE;

VC address is 5 for default signaling channel (and VP=0); and VC address is 16 for default ILMI channel.

The 8 circuits can operate in parallel and cells may be subjected to the match test to determine whether or not the cell should be stripped out of the stream. Preferably, this function is required for composite ingress streams on the ABCU card 22. In the case of the ingress stream from PHY ports, a routing tag is appended to the cell to identify the port the cell came from. Each of the addresses supported by the MegaSLAM are preferably programmed to support any combination of 32 bits. For example, five of these registers could be provisioned for the five VC addresses listed herein, leaving three unused registers, which, for example, could be used for a peer-to-peer link communication protocol or VCC F5 OAM flows.

One of the circuits preferably provides an additional feature to evaluate the content of an OAM cell type and function (TBD) field and, based on the content of these fields, forward the cell to the daisy chained link. At the same time, this circuit can forward the same cell to the local CPU. This feature provides a point-to-multipoint connection over the daisy chained links. This is useful for the control cells that are being exchanged between switching units 104.

3.9.1 F4 and F5 OAM Cell Flows

Switching subsystem 100 is preferably considered a single network segment. Segment flows are terminated only in the last switching unit 104n. Switching subsystem 100 will generate and terminate F4 OAM cell flows. Hardware VCI/VPI address mapping function will strip these OAM cells out of the cell stream and pass them to the local CPU. The hardware also checks the CRC-10 and provide CRC indication to the CPU. A hardware interface primitive Enable_F4_flows preferably performs the following function: when true, the hardware strips F4 flows out of the cell stream. The CPU cell TX Fifo can, under software control, at any time queue a cell for transmission on any outbound composite link (or bus), therefore no primitive is needed to support sending F4 OAM cells.

An injection FIFO is provided for each of the composite egress streams on the ABCU card 22. This FIFO provides at least double buffering for two cells that can be injected into a composite stream. This FIFO takes priority over other streams. A software scheduler controls the rate of CPU injected cells. The CPU software will provide the drivers required to service these cell streams.

The system does not interfere with the in band F5 flows. The F5 flows will transparently pass through the switching subsystem 100. They are expected to be terminated in the CPE device.

In embodiments where the CPE does not support some of the OAM flows, VC or VP OAM flows may be generated as a proxy for the CPE as a provisioning option.

3.10 Performance Monitoring and Fault localization

Switching subsystem 100 preferably provides both traditional physical layer procedures and ATM cell layer procedures. In some cases, both procedures may not be required and a simpler, more cost effective solution results.

Loopbacks may be provided for the full payload (PHY level), the virtual path payload, and maybe the virtual circuit payload. In addition, it may make sense to inject a small amount of overhead into the stream to do a continuous performance monitoring function. This overhead in the cell domain could be looped back at the CPE.

3.10.1 ATM Cell level Procedures

Switching subsystem 100 provides ATM performance monitoring procedures at external interfaces and the connection between the ABCU card 22 and the daisy chained ABCU card 22. For the drops, it is performed by the drop PHY and for the ABCU card 22 interfaces. The following parameters, for example, may be measured:

- CER, cell error ratio
- CLR, cell loss ratio
- CMR, cell miss-inserted rate
- SECBR, severely errored cell block ratio
- Number of cells with parity error on transmit
- Number of discard cells due to double HEC error
- Number of corrected single HEC error Cells
- OAM cells with CRC-10 error

C. FUNCTIONAL OPERATION

FIGS. 9-14 provide functional operation perspective of switching subsystem 1100. Referring to FIG. 9, a distributed telecommunications switching subsystem 1100 is shown. Switching subsystem 1100 comprises a plurality of switching units 1102, 1104, and 1106, referred to as channel banks. Each channel bank provides data and/or voice communication services to a plurality of customer premises equipment (CPE) units 1108. A primary channel bank 1102 communicates with a data packet switch 1110, such as an asynchronous transfer mode (ATM) switch 1110, which in turn communicates with a telecommunications network 1112. ATM switch 1110 may, for example, be located at a telephone company central office one or more intermediate channel banks 1104 may be positioned between primary channel bank 1102 and a terminating channel bank 1106.

In the preferred embodiment described herein, the primary function of switching subsystem 1100 is to route data packets in the well known ATM cell format from ATM switch 1110 to individual CPE units 1108 and to carry ATM cells from CPE units 1108 to ATM switch 1110. Together, ATM switch 1110 and switching subsystem 1100 provide communication paths between CPE units 1108 and one or more destinations in telecommunications network 1112. It will be understood that the distributed telecommunications switching subsystem and method described herein may also be employed to route digital or analog information encoded in other formats, such as Transmission Control Protocol/Internet Protocol data packets.

In the following discussion, ATM cells being sent from ATM switch 1110 through switching units 1102, 1104, and 1106 to CPE units 1108, or any other destination in switching subsystem 1100, will be referred to as traveling in the downstream direction. Any cells sent from CPE units 1108 through switching units 1102, 1104, and 1106 to ATM switch 1110 will be referred to as traveling in the upstream direction.

Primary channel bank 1102 communicates with ATM switch 1110 by means of communication line 1114 which

carries ATM cells downstream from ATM switch 1110 to primary channel bank 1102. Primary channel bank 1102 also communicates with ATM switch 1110 by means of communication line 1116 which carries cells upstream from primary channel bank 1102 to ATM switch 1110. In the preferred embodiment, communication lines 1114 and 1116 are fiber optic cables capable of carrying data at a standard OC-3 data rate.

Primary channel bank 1102 comprises a controller 1118 referred to as an ATM bank controller unit (ABCU) and a plurality of subscriber interface cards 1120 referred to as asymmetric digital subscriber line (ADSL) cards. Controller 1118 transmits cells downstream to subscriber interface cards 1120 on a shared high speed cell bus 1126. Subscriber interface cards 1120, 1122 and 1124 transmit cells upstream to controller 1118 via serial bus interface (SBI) lines 1128, 1130, and 1132, respectively.

Controller 1118 sends cells downstream to intermediate channel bank 1104 via communication line 1134, and receives cells traveling upstream via communication line 1136. Communication lines 1134 and 1136, like lines 1114 and 1116, are preferably fiber optic cables capable of carrying data at the standard OC-3 data rate.

Downstream intermediate channel banks 1104 and terminating channel bank 1106 are similar in structure to primary channel bank 1102, each having a controller 1138 and 1140, respectively, and a plurality of subscriber interface cards 1120. Some differences in functionality among the channel banks will become apparent from the description to follow.

Intermediate channel bank 1104 may be directly coupled to terminating channel bank 1106 by communication lines 1142 and 1144. Alternatively, one or more channel banks may be situated between intermediate channel bank 1104 and terminating channel bank 1106 in a "daisy chain" arrangement, with each channel bank being connected to the previous one by communication lines, as shown. Switching subsystem 1100 preferably comprises up to nine channel banks. Regardless of the number of channel banks in switching subsystem 1100, terminating channel bank 1106 is the last channel bank in the chain.

Each channel bank 1102, 1104, 1106 may include up to 60 subscriber interface cards 1120, with each subscriber interface card 1120 communicating with up to four separate CPE units 1108. The communication with CPE units 1108 is asymmetric, with an exemplary data rate of six million bits per second (6 Mbps) supplied to the customer and 640 Kbps received from the customer. The type of service provided to the customer may be plain old telephone service (POTS), data service, or any other telecommunications service, and may or may not include a minimum cell rate (MCR) guaranteed for the customer's upstream data communications.

Generally, switching subsystem 1100 will be oversubscribed in the upstream direction, meaning that the cumulative peak cell rate (PCR) which may be transmitted by the customers exceeds the maximum rate at which switching subsystem 1100 may transmit cells to ATM switch 1110. Control methods that allow switching subsystem 1100 to provide adequate service to oversubscribed customers will be discussed more fully below.

Referring to FIG. 10, a functional block diagram of an upstream controller 1150 in accordance with the invention is shown. Controller 1150 may be implemented in switching subsystem 1100 as controller 1118 or 1138, or as a controller for another intermediate channel bank situated between intermediate channel bank 1104 and terminating channel bank 1106.

Controller 1150 receives cells traveling downstream from ATM switch 1110 or another controller in an upstream channel bank via fiber optic cable 1152 and send cells upstream from a downstream channel bank via fiber optic cable 1154. Controller 1150 sends cells downstream to another channel bank via fiber optic cable 1156 and receives cells upstream from a downstream channel bank via fiber optic cable 1158.

Controller 1150 transmits appropriate cells downstream to subscriber interface cards 1120 on a shared high speed cell bus 1160. When a large number of subscriber interface cards 1120 are serviced by controller 1150, high speed cell bus 1160 may comprise a plurality of separate lines, each carrying the same high speed signal to a separate set of subscriber interface cards 1120. For example, in a configuration with 60 subscriber interface cards being serviced by controller 1150, high speed cell bus 1160 may comprise three separate lines, each connected to 20 subscriber interface cards 1120, but each carrying cells addressed to all of the subscriber interface cards 1120.

Each subscriber interface card 1120 sends cells upstream to controller 1150 via a separate subscriber bus interface line 1162, 1164, or 1166. In addition to carrying ATM traffic, subscriber bus interface lines 1162, 1164, and 1166 may also carry telephone traffic from POTS subscribers. In that case, the POTS traffic may be separated out from the ATM traffic and processed by other equipment not shown. This separation occurs before the processing of ATM cells described herein. The downstream communication of POTS traffic to subscriber interface cards 1120 may occur on lines other than high speed cell bus 1160.

Buffers 1168, 1170 and 1172 receive ATM signals on subscriber bus interface lines 1162, 1164 and 1166, respectively, and store the received data until one or more complete cells are received. The cells are then passed on to an internal switching controller 1174, which comprises an address storage system 1176, a processor 1178, and a switch 1180.

Address storage system 1176 stores a list of addresses corresponding to the CPE units 1108 serviced by controller 1150. In the preferred embodiment, each address identifies a virtual path and virtual circuit for a CPE unit 1108 in an addressing format well known to those skilled in the art of ATM communications. However, it will be appreciated that other addressing systems, such as Internet Protocol addressing, may be used to identify cell destinations both within and outside switching subsystem 1100.

Incoming signals on fiber optic cables 1152 and 1158 are converted to electrical signals by fiber optic couplers 1182 and 1184, respectively. The converted signals are transmitted to internal switching controller 1174.

Internal switching controller 1174 transmits cells downstream to a downstream channel bank via fiber optic cable 1156. To accomplish this, cells are transmitted to a plurality of first in first out (FIFO) buffers or queues 1186 and 1188 controlled by a scheduler 1190. When triggered by scheduler 1190, each queue 1186 or 1188 dequeues one or more cells, transmitting the cells to a fiber optic coupler 1192 which converts the data signals to optical signals for transmission over fiber optic cable 1156.

Likewise, internal switching controller 1174 transmits cells upstream to an upstream channel bank or ATM switch 1110 via fiber optic cable 1154. To accomplish this, cells are transmitted to a plurality of FIFO queues 1194, 1196 and 1198 controlled by a scheduler 1200. When triggered by scheduler 1200, each queue 1194, 1196, or 1198 dequeues

one or more cells, transmitting the cells to a fiber optic coupler 1202 which converts the data signals to optical signals for transmission over fiber optic cable 1154.

In operation, controller 1150 receives downstream ATM cells from an upstream channel bank or ATM switch 1110 on fiber optic cable 1152. Processor 1178 compares the address portion of a received cell to the list of addresses stored in address storage system 1176. If a match is found, then switch 1180 transmits the cell to the subscriber interface cards 1120 associated with controller 1150 on shared high speed cell bus 1160.

All of the subscriber interface cards 1120 associated with controller 1150 check the address of the transmitted cell carried over high speed cell bus 1160 and compare it to their internal address lists. Only the subscriber interface card 1120 servicing the CPE unit 1108 to which the cell is addressed reacts to receipt of the cell. All other subscriber interface cards ignore the cell.

Returning to controller 1150, if the address of the cell did not match any of the addresses stored in address storage system 1176, then processor 1178 compares the address of the cell to a processor address to determine whether the cell is a control cell addressed to processor 1178. If the address matches the processor address, then the control cell is processed by processor 1178 in a manner to be described below.

If the cell address does not match any address for controller 1150, then the cell is sent by switch 1180 to a bypass queue 1186. When bypass queue 1186 receives a cell, it sends a ready signal to scheduler 1190 which coordinates transmissions over fiber optic cable 1156 to a next downstream channel bank. When scheduler 1190 sends a transmit signal to bypass queue 1186, the cell is transmitted to coupler 1192 and onto fiber optic cable 1156.

Processor 1178 may also generate control cells for transmission to downstream channel banks, as will be described more fully below. When processor 1178 generates such a cell, the cell is passed by switch 1180 to CPU queue 1188, which transmits a ready signal to scheduler 1190. Scheduler 1190 preferably controls both bypass queue 1186 and CPU queue 1188 to ensure that CPU queue 1188 receives higher priority than bypass queue 1186. This priority scheme may be implemented in a variety of ways. For example, bypass queue 1186 may be allowed to dequeue a cell only when CPU queue 1188 is empty. Because the frequency of control cells is low, this priority scheme does not significantly impede downstream traffic.

It will be appreciated by those skilled in the art that the downstream cell switching process executed by controller 1150 differs from that of a telecommunications switching system arranged in a tree structure. Rather than storing addresses for all customers located downstream of controller 1150, address storage system 1176 only stores addresses corresponding to the customers directly serviced by controller 1150. Any cell having an unrecognized address is passed downstream to another controller for processing. This allows for a smaller address storage system 1176 and faster address processing in controller 1150.

In the upstream direction, controller 1150 receives ATM cells from downstream channel banks on fiber optic cable 1158. Processor 1178 compares the address portion of a received cell to its own address to determine whether the cell is a control cell addressed to processor 1178. If the address matches the processor address, then the control cell is processed by processor 1178 in a manner to be described below.

If the cell address does not match the processor address, then the cell is sent by switch 1180 to a bypass queue 1194. When bypass queue 1194 receives a cell, it sends a ready signal to scheduler 1200, which coordinates transmissions over fiber optic cable 1154. When scheduler 1200 sends a transmit signal to bypass queue 1194, the cell is transmitted to coupler 1202 and onto fiber optic cable 1154.

If controller 1150 is implemented in a downstream channel bank, i.e. a channel bank other than primary channel bank 1102, then processor 1178 may also generate control cells for transmission to upstream channel banks, as will be described more fully below. When processor 1178 generates such a cell, the cell is passed by switch 1180 to a CPU queue 1196, which transmits a ready signal to scheduler 1200. When scheduler 1200 sends a transmit signal to CPU queue 1196, the control cell is transmitted to coupler 1202 and on to fiber optic cable 1154.

Cells are received from the local CPE units 1108 serviced by controller 1150 on subscriber bus interface lines 1162, 1164, and 1166. As previously noted, controller 1150 may receive cells from up to 60 subscriber bus interface lines. Processor 1178 checks the address portion of each cell to determine whether the cell is addressed to processor 1178 itself or to a valid upstream destination.

The subscriber interface cards 1120 controlled by controller 1150 may, for example, send status feedback cells to processor 1178 indicating whether traffic congestion is occurring in the subscriber interface cards 1120. Processor 1178 processes these status feedback cells accordingly.

Other cells addressed to valid upstream destinations are transmitted by switch 1180 to ingress queue 1198. Scheduler 1200 controls bypass queue 1194, CPU queue 1196, and ingress queue 1198 to implement a selected priority scheme. In the preferred embodiment, CPU queue 1196 receives the highest priority, bypass queue 1194 receives the next priority, and ingress queue 1198 receives the lowest priority. As with scheduler 1190, this priority scheme may be implemented in a variety of ways. For example, ingress queue 1198 may be allowed to dequeue a cell only when CPU queue 1196 and bypass queue 1104 are both empty. Because the frequency of control cells is low, this priority scheme does not significantly impede upstream traffic.

In an alternative embodiment of controller 1150, ingress queue 1198 actually comprises 16 separate ingress queues, as shown in FIG. 11. Each ingress queue 1198a-1198p is assigned a separate priority. As in the previous embodiment, a priority scheme is enforced by scheduler 1200.

The priority scheme allows each queue to provide different classes of service to customers. For example, each ingress queue may receive cells belonging to one of the well-known ATM cell traffic classes, as illustrated in FIG. 11. In this example, ingress queues 1198a through 1198h are spare queues, ingress queue 1198i receives unspecified bit rate (UBR) traffic with fair performance, ingress queue 1198j receives UBR traffic with good performance, ingress queues 1198k, 1198l and 1198m receive variable bit rate (VBR) traffic with guaranteed minimum cell rates of 64 Kbps, 128 Kbps and 256 Kbps, respectively, ingress queue 1198n receives VBR traffic with guaranteed 100% cell throughput, ingress queue 1198o receives real-time variable bit rate (VBR) traffic, and ingress queue 1198p receives constant bit rate (CER) traffic.

In this embodiment, internal switching controller 1174 assigns cells to different ingress queues according to the origin of each cell. Customers serviced by switching subsystem 1100 select in advance a class of service they would

like to receive, with higher priority traffic classes and guaranteed minimum throughputs being more expensive than low priority and/or oversubscribed service. Each customer's cells are then sent by internal switching controller 1174 to the appropriate ingress queue 1198a through 1198p.

Scheduler 1200 and processor 1178 are programmed to dequeue upstream queues 1194, 1196 and 1198 according to a predetermined priority scheme. The optimal priority scheme to implement depends on a number of situation-specific factors, such as the number of ingress queues, the classes of service offered, the oversubscription ratio, and predicted traffic load statistics. However, certain guidelines must be followed. For example, ingress queue 1198k must be allowed to dequeue cells often enough to achieve the minimum throughput of 64 Kbps.

The priority scheme implemented by scheduler 1200 and processor 1178 may vary with the level of traffic congestion in controller 1150. For example, any ingress queues 1198a through 1198p that are not empty may be dequeued in a round robin fashion unless the traffic congestion in controller 1150 reaches a threshold level, at which point the minimum cell rate guarantees for some ingress queues require a preferential dequeuing process to be implemented.

It will be appreciated that the various elements of controller 1150, excluding fiber optic couplers 1182, 1184, 1192, and 1202, generally perform data storage and signal processing functions, and may therefore be implemented as hardware, firmware, software, or some combination thereof.

Referring to FIG. 12, a functional block diagram of controller 1140 is shown. Controller 1140 is similar in structure to controller 1150 described above in connection with FIG. 10. However, because controller 1140 controls terminating channel bank 1106 in switching subsystem 1100, controller 1140 does not receive or transmit cells to any downstream channel banks. For the purposes of this description only, it will be assumed that switching subsystem 1100 comprises only three channel banks and that controller 1140 therefore communicates directly with controller 1138.

Signals traveling downstream on fiber optic cable 1142 are converted to electrical signals by fiber optic coupler 1204. The converted signals are transmitted to internal switching controller 1206.

Internal switching controller 1206 transmits cells to controller 1138 via fiber optic cable 1144. To accomplish this, cells are transmitted to a plurality of FIFO queues 1220 and 1222 controlled by a scheduler 1224. When triggered by scheduler 1224, each queue 1220 or 1222 dequeues one or more cells, transmitting the cells to a fiber optic coupler 1226 which converts the data signals to optical signals for transmission over fiber optic cable 1144.

For downstream operation, controller 1140 receives ATM cells from upstream channel bank 1104 on fiber optic cable 1142. A processor 1208 of internal switching controller 1206 compares the address portion of a received cell to the list of addresses stored in address storage system 1210. If a match is found, then a switch 1212 transmits the cells to the subscriber interface cards 1120 associated with controller 1140 on shared high speed cell bus 1214.

If the address of the cell does not match any of the addresses stored in address storage system 1210, then processor 1208 compares the address of the cell to its own address to determine whether the cell is a control cell addressed to processor 1208. If the address matches the processor address, then the control cell is processed by processor 1208 in a manner to be described below.

If the cell address does not match the processor address, then the cell has failed to match any of the addresses serviced by switching subsystem 1100. At this point, the cell is deemed a mis-inserted cell and is processed by processor 1208 which may gather statistics on such cells. Mis-inserted cells may, for example, indicate that an unauthorized party is attempting to receive service from switching subsystem 1100.

In the upstream direction, cells are received from the local CPE units 1108 serviced by controller 1140 on subscriber bus interface lines 1215, 1216, and 1218. As previously noted, controller 1140 may receive cells from up to 60 subscriber bus interface lines. Processor 1208 checks the address portion of each cell to determine whether the cell is addressed to processor 1208 itself or to a valid upstream destination.

Cells addressed to valid upstream destinations are transmitted by switch 1212 to ingress queue 1220. Processor 1208 may also generate control cells for transmission to upstream channel banks, as will be described more fully below. When processor 1208 generates such a cell, the cell is passed by switch 1212 to a CPU queue 1222.

A scheduler 1224 controls CPU queue 1222 and ingress queue 1220 to implement the selected priority scheme as previously described. In the preferred embodiment, CPU queue 1222 receives higher priority than ingress queue 1220. Because the frequency of control cells is low, this priority scheme does not significantly impede upstream traffic.

From the foregoing description, it will be appreciated that switching subsystem 1100 provides distributed telecommunications switching which features several advantages over a traditional tree structure. Each channel bank only stores a limited number of addresses pertaining to customers directly serviced by the channel bank, and is effectively independent of the other channel banks in the system.

In addition to simplifying the setup for switching subsystem 1100, the modularity of the system allows expansion of service with minimal modification to the existing structure. When a set of new customers is to be serviced, a new channel bank may be added into switching subsystem 1100. The new channel bank may be programmed with the addresses of the new customers, while the cell processing methods and address storage for other channel banks remain unaffected.

The channel banks in switching subsystem 1100 may also be located remotely from one another without significant degradation in service. This allows customers in different locations to be "close to the switch," decreasing access times for the customers and improving service.

Because switching subsystem 1100 is oversubscribed in the upstream direction, some control system must be implemented to ensure uniformity in quality of service for customers throughout switching subsystem 1100. For example, if upstream bypass queue 1194 in controller 1118 receives higher priority than ingress queue 1198, then CPE units 1108 serviced by channel bank 1102 may be effectively blocked from access to ATM switch 1110 due to heavy upstream traffic. An upstream flow control system must be implemented to ensure fairness throughout switching subsystem 1100.

Two different upstream flow control systems will be described herein. Although these control systems are presented as mutually exclusive alternatives, it will be appreciated that variations and combinations of these two control schemes may be implemented without departing from the spirit and scope of the invention.

Referring to FIG. 13, the operation of the first upstream flow control system is illustrated. In this control system, controller 1118 in channel bank 1102 periodically initiates a control loop by generating a control cell 1230. In general terms, the control cell performs two functions: providing control information to each channel bank in switching subsystem 1100 and triggering a status feedback cell 1232 that provides information to controller 1118 concerning the cell traffic congestion at each channel bank. The control cell is preferably generated only when controller 1118 is not experiencing high traffic congestion levels in the upstream direction so that the returning status feedback cell 1232 will not contribute to upstream traffic congestion.

An exemplary format for control cell 1230 is shown in Table A. This cell follows a standard ATM Organization, Administration and Maintenance (OAM) cell format. Thus, octets 1 through 5 include standard ATM header information and octet 6 includes OAM and function type information, which identifies the cell as a control cell.

Octets 7 and 8 contain a control command word which sets the length or interval of a control cycle, expressed as a number of cells. Thus, if the control command word has a value of 128, then a control cycle will be deemed to constitute an interval of 128 cells in the upstream flow. Every 128 cells then constitutes a separate control cycle.

TABLE A

Octet	Function
1-5	standard ATM header
6	4 bits OAM type 4 bits Function type
7-8	Control command word - contains length of control cycle in cell times
9-24	8 words of 16 bits contain the credit allowance for each of the 8 daisy chained channel banks octets 9 and 10 are for the first channel bank octets 23 and 24 are for the last channel bank
25-46	spare
47-48	6 bits reserved 10 bits for CRC-10

Octets 9 through 24 contain up to eight credit allowance words of 16 bits each. One credit allowance word is included for each downstream channel bank in switching subsystem 1100. Thus, for example, if channel banks 1102, 1104 and 1106 were the only channel banks in switching subsystem 1100, then octets 9 through 12 would contain one credit allowance word each for channel banks 1104 and 1106, while octets 13 through 24 will remain empty. Since the credit allowance control cell is generated by controller 1118 of primary channel bank 1102, the credit allowance for primary channel bank 1102 is processed directly by its processor 1178 and need not be placed within the credit allowance control cell.

The credit allowance word for a channel bank indicates the number of cells in a control cycle that are allotted to that channel bank for transmission upstream. For example, if the control cycle length is 128 cells, and the credit allowance word for channel bank 1104 has a value of 43, then controller 1138 may transmit 43 cells upstream on fiber optic cable 1136 during the next 128-cell interval.

This credit-based upstream flow control is implemented by processor 1178 shown in FIG. 10. Thus, processor 1178 maintains a counter (not explicitly shown) which is decremented by one every time processor 1178 through scheduler

61

1200 dequeues a cell from ingress queue 1198. When the counter reaches zero, no more cells are dequeued from ingress queue 1198 until the next control cycle.

Returning to Table A, Octets 25 through 46 of the control cell are unused. Octets 47 and 48 include 10 bits used for a cyclical redundancy check (CRC) of the control cell while the other six bits remain unused.

When a control cell is generated by controller 1118, the control cell is passed to CPU queue 1188 for transmission downstream to controller 1138. Controller 1138 receives the control cell and reads octets 7 through 10 to determine the length of the control cycle and the credit allowance for channel bank 1104. Controller 1138 then passes the control cell downstream, unmodified.

Likewise, each controller downstream receives the control cell, reads its own credit allowance, and passes the control cell further downstream, as illustrated in FIG. 13. Controller 1140 in channel bank 1106 discards the control cell after reading it.

Controller 1140 is programmed to respond to the receipt of a control cell by generating a status feedback cell 1232. This cell is passed upstream, with cell traffic congestion information being written into the status feedback cell by each controller in switching subsystem 1100. When the cell reaches controller 1118 in channel bank 1102, the status feedback information is read and the cell is discarded.

An exemplary format for status feedback cell 1232 is shown in Table B. Like control cell 1230 described above, the status feedback cell follows the standard OAM format. Thus, octets 1 through 5 include standard ATM header information and octet 6 includes OAM and function type information which identifies the cell as a status feedback cell.

TABLE B

Octet	Function
1-5	standard ATM header
6	4 bits OAM type 4 bits Function type
7-12	8 words of 16 bits contain the status for each of the 8 daisy chained channel banks octets 7 and 8 are for the first channel bank octets 21 and 22 are for the last channel bank
23-44	spare
45-46	Number of cells in upstream bypass queue or last Release Two shelf
47-48	6 bits reserved 10 bits for CRC-10

Octets 7 through 22 contain up to eight status feedback words of 16 bits each. One status feedback word appears for each channel bank in switching subsystem 1100. Thus, for example, if channel banks 1102, 1104 and 1106 are the only channel banks in switching subsystem 1100, then octets 7 through 10 will contain one credit allowance word each for channel banks 1104 and 1106, while octets 11 through 22 will remain empty. The feedback status for primary channel bank 1102 is directly handled by its processor 1178 and thus does not be inserted into the status feedback control cell.

The status feedback word for each channel bank identifies the current traffic congestion level at the channel bank. It will be appreciated that various formats may be used to identify traffic congestion levels. In the preferred embodiment, one of four traffic congestion levels is ascribed to ingress queue 1198.

62

In the embodiment shown in FIG. 11, in which ingress queue 1198 comprises 16 separate ingress queues, each with its own priority level, a separate traffic congestion level is ascribed to each priority level group of ingress queues. The status feedback word format for this embodiment is illustrated in Table C.

TABLE C

Bit	Function
0-9	free list
10-11	congestion state for lowest priority group of queues 0 = level 0 1 = level 1 2 = level 2 3 = level 3
12-13	congestion state for second to lowest priority group of queues 0 = level 0 1 = level 1 2 = level 2 3 = level 3
14-15	congestion state for third to lowest priority group of queues 0 = level 0 1 = level 1 2 = level 2 3 = level 3

Generally, the traffic congestion level for a queue is determined by reference to the buffer space allotted for the queue. The higher the amount of allotted buffer space being utilized by the queue, the higher the traffic congestion level for the queue.

The threshold congestion levels which quantitatively define the four traffic congestion levels vary from queue to queue according to variables such as queue size, free buffer space, anticipated queue traffic patterns, and in some cases the rate of decrease of free buffer space. However, in general terms, Level 0 represents a normal or uncongested state, Level 1 represents a near congestion state, Level 2 represents a congestion imminent state, and Level 3 represents a congested state.

These congestion levels may be used not only to provide feedback to controller 1118, but also to regulate cell processing within a downstream controller 1150. For example, at Level 0, cell handling may proceed normally. At Level 1, processor 1178 may begin implementing congestion control measures such as early packet discard (EPD), partial packet discard (PPD) and/or restricting the cell flow rate to ingress queues 1198a through 1198p on a queue-by-queue basis. At Levels 2 and 3, these congestion control measures may be implemented in a progressively severe manner.

Referring to Table C, bits 0 through 9 of the status feedback word give the total free buffer space available for the ingress queues. Bits 10 and 11 give the traffic congestion level for the lowest priority group of queues, which may be, for example, queues 1198i and 1198j. Bits 12 and 13 give the traffic congestion level for the second lowest priority group of queues, which may be, for example, queues 1198k through 1198n. Bits 14 and 15 give the traffic congestion level for the third lowest priority group of queues, which may be, for example, queues 1198o and 1198p.

Controller 1140, and more particularly processor 1208 therein, originally generates status feedback cell 1232, with octets 7 and 8 containing the status feedback word for channel bank 1106. The status feedback cell is then passed upstream from controller to controller, as illustrated in FIG.

13, with each controller writing its own status feedback word into the appropriate two octets of the status feedback cell. When controller 1118 in channel bank 1102 receives status feedback cell 1232, the cell is routed to processor 1178, which utilizes the traffic congestion information contained in status feedback cell 1232, as well as traffic congestion information from controller 1118 itself, to determine an appropriate credit distribution to be included in the next control cell 1230.

This process is repeated periodically during the operation of switching subsystem 1100. Each control cell 1230 generated by processor 1178 includes a credit distribution for the downstream channel banks based upon information from the previous status feedback cell 1232. Processor 1178 also assigns credits for controller 1118, but this information remains internal to controller 1118 and is not included in control cell 1230.

In this control system, controller 1140 in channel bank 1106 launches cells upstream at will from CPU queue 1222, and utilizes its assigned credits to launch cells from ingress queue 1220. During intervals when CPU queue 1222 and ingress queue 1220 are either empty or not allowed to launch cells upstream, controller 1140 launches a steady stream of empty or unassigned cells. Each upstream controller receives the stream of empty cells and replaces empty cells with cells from its own queues in accordance with its priority scheme and credit allowance.

In the case where the number of empty cells transmitted upstream to controller 1118 in channel bank 1102 exceeds the number of credits assigned to channel bank 1102, controller 1118 may be programmed to dequeue cells from its ingress queues in excess of its credit allowance. This flexibility ensures maximum utilization of upstream bandwidth resources.

Referring to FIG. 14, the operation of the second upstream control system is illustrated. In this system, bandwidth on the upstream fiber optic cables is pre-assigned according to class of service or queue priority. This differs from the first embodiment, in which bandwidth is assigned for each channel bank, with a local scheduler in each controller making dequeuing decisions to allocate bandwidth for queues with different priorities. In the second embodiment, queues having the same priority, regardless of the channel bank in which they are located, may compete for the bandwidth assigned to that queue class.

In this control system, controller 1140 in channel bank 1106 generates a continuous stream of cells 1234, some or all of which are marked as reserved for particular queue classes. This marking occurs in the cell header in the location that usually contains address information. More specifically, the virtual path indicator is replaced with a unique code identifying the cell as reserved. The virtual circuit indicator is replaced with an identification of the queue class for which the cell is reserved.

A queue class may be a simple priority or traffic class designation. For example, a CPU queue such as queue 1188 in each controller in switching subsystem 1100 may be designated as Queue Class One. Thus, a Queue Class One reserved cell sent upstream from controller 1140 will be used by the first controller that has a non-empty CPU queue 1188.

Queue classes may also provide further subdivision of queues. For example, if switching subsystem 1100 comprises nine channel banks, Queue Class One may be used to designate CPU queues in the lower three channel banks, Queue Class Two may be used to designate CPU queues in

the middle three channel banks, and Queue Class Three may be used to designate CPU queues in the upper three channel banks. Likewise, a queue class may be used to designate a selected queue or set of queues in one particular channel bank.

Queue classes may also designate groups of queues servicing different traffic classes. For example, one queue class may be used to designate all queues carrying "concentrated" or oversubscribed cell traffic, such as ABR and UBR queues, while another queue class may be used to designate all queues carrying non-concentrated traffic, such as VBR and CBR queues.

In each controller, internal switching controller 1174 is programmed with the queue class designations of each upstream queue 1194, 1196 and 1198. Thus, when a reserved cell for a queue class is received on fiber optic cable 1158, processor 1178 cooperates with scheduler 1200 to ensure that, if a non-empty queue belonging to that queue class exists in controller 1150, then a cell is dequeued from the non-empty queue. Otherwise, the reserved cell is passed upstream without modification.

If the reserved cell reaches controller 1118, it must be replaced with a queued cell or an unassigned cell. This is because the non-standard format used to designate reserved cells will not be recognized by ATM switch 1110. Reserved cells must therefore be removed from the stream before reaching ATM switch 1110.

In an exemplary priority scheme, illustrated in FIG. 14, controller 1140 of terminating channel bank 1106 generates a repeating sequence 1234 of 1000 cells. In this sequence, 50 of the cells, represented by cell 1234a, are reserved for concentrated traffic, while 100 cells, represented by cell 1234e, are reserved for non-concentrated (CBR and VBR) traffic. The remaining cells are generally unassigned, i.e. empty and not reserved, as illustrated by cells 1234b and 1234c.

Channel bank 1106 not only creates the reserved cell distribution, but also takes part in the cell reservation system as a "consumer" of upstream bandwidth. Thus, controller 1140 dequeues cells from its queues 1220 and 1222 in place of some of the unassigned cells and/or reserved cells before launching the cells upstream, as illustrated by cell 1234d in FIG. 14.

In this priority scheme, when an unassigned cell is received at a controller 1150, processor 1178 and scheduler 1200 implement an internal priority scheme that gives non-concentrated traffic queues priority over concentrated traffic queues. However, five percent of the cells received are marked as reserved for concentrated traffic, ensuring that concentrated traffic queues are allowed to dequeue a minimum number of cells even when non-concentrated traffic is heavy.

Thus, referring to FIG. 14, channel bank 1105f receives the cell stream 1234 and dequeues a cell 1234f from a concentrated traffic queue to take the place of reserved cell 1234a. Channel bank 1105e dequeues two cells 1234g and 1234h from non-concentrated traffic queues to replace unassigned cell 1234b and reserved cell 1234e, respectively. For channel banks upstream of channel bank 1105e, only one unassigned cell 1234c remains to be replaced by a dequeued traffic cell.

To ensure that the supply of reserved cells is not completely exhausted before reaching upstream channel banks such as primary channel bank 1102 and intermediate channel banks 1104, fairness assurance procedures may also be built into this control system. For example, scheduler 1200 and/or

processor 1178 in each controller may be programmed to limit the rate at which any particular queue or group of queues may dequeue cells upstream.

Another method for ensuring fairness is to implement a queue class system in which queues in the upstream channel banks such as primary channel bank 1102 and intermediate channel banks 1104 may be designated separately from the downstream channel bank queues as previously described. Then, controller 1140 in channel bank 1106 may reserve a minimum number of cells specifically for the queues in specific upstream channel banks.

Thus, it is apparent that there has been provided, in accordance with the present invention, a distributed telecommunications switching subsystem and method that satisfy the advantages set forth above. Although the present invention has been described in detail, it should be understood that various changes, substitutions, and alterations readily ascertainable by one skilled in the art can be made herein without departing from the spirit and scope of the present invention as defined by the following claims.

4. Acronyms

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ADSL	Asymmetrical Digital Subscriber Line
AM	Amplitude Modulation
ATM	Asynchronous Transfer Mode
BB	Broadband
BCC	Broadcast Channel Selection
BCST	Broadcast
BFB	Broadband Fiber Bank
BFD	Buffer descriptor
BORSH	Battery, Overvoltage, Ringing, Supervision,
T	Hybrid, Test: the functions of the POTS line circuit
BPS	Bank Power Supply
BFT	Central Control of Narrowband ONU
CAC	Connection Admission Control
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CES	Circuit Emulation Service
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CO	Central Office
COM	Continuation of Message
COT	CO Terminal
CPE	Customer Premises Equipment
CRU	Cell routing Unit
CTTH	Coax To The Home
DCS	Digital Cross-connect System
DEHN	Digital Home Network
DS	Delivery System
DVB	Digital Video Broadcast
EFCI	Explicit Forward Congestion Indication
EOM	End Of Message
EFD	Early Packet Discard
ESC	End Service Consumer
ESF	Extended Super Frame
ESP	End Service Provider
FBIU	Fiber Bank Interface Unit
FTTH	Fiber To The Home
GCRA	Generic Cell Rate Process
HAN	Home Access Network
HDT	Host Digital Terminal
HEC	Header Error Check
HFC	Hybrid Fiber Coax
IOF	Inter-Office Facilities
ISP	Internet Service Provider
L1GW	Level 1 Gateway
L2GW	Level 2 Gateway
LDS	Local Digital Switch
LSB	Least Significant Bit
LSBB	Litespan Broadband
LTM	Litespan Broadband Traffic Management
MOD	Movie On Demand

-continued

MSB	Most Significant Bit
NIU	Network Interface Unit
NNI	Network to Network Interface
NMS	Network Management System
NOD	Network Ownership Decoupling
NT	Network Termination
NTM	Network side Traffic Management
O/E	Opto-Electrical conversion
OA&M	Operations, Administration and Maintenance
OAM	Operation and Maintenance Cell
OC-n	Optical Carrier hierarchy
OLU	Optical Line Unit
ONU	Optical Network Unit
ORM	Optical Receiver Module
PDU	Packet Data Unit
PHS	Per Home Scheduler
PHY	Physical Layer (ATM protocol stack)
POTS	Plain Old Telephone Service
PPD	Partial Packet Discard
PPV	Pay Per View
PRI	Priority - arbiter or scheduler
PWR	Power
QD	Queue Descriptor
QoS	Quality of Service
RM	Resource Management Cell
RRS	Round Robin Select - arbiter or scheduler
RSU	Remote Switching Unit
SAM	Service Access Mux
SDV	Switched Digital Video
SPS	Service Provider System
STB	Set-Top Box
STU	Set-Top Unit
TC	Transmission Convergence (ATM protocol stack layer)
TDM	Time Division Multiplex
TP	Twisted Pair
TFTTH	Twisted Pair To The Home
TSI	Time Slot Interchange
TTD	Transmission Technology Decoupling
UNI	User Network Interface
UPC	Usage Parameter Control, (i.e. policing)
UPI	User Premises Interface
VASP	Value Added Service Provider
VBR	Variable Bit Rate
VC	Virtual Channel
VCD	Virtual Circuit Descriptor
VCI	Virtual Channel Identifier
VF	Voice Frequency
VIP	Video Information Provider
VIU	Video Information User
VOD	Video On Demand
VP	Virtual Path
VPI	Virtual Path Identifier

A few preferred embodiments have been described in detail hereinabove. It is to be understood that the scope of the invention also comprehends embodiments different from those described, yet within the scope of the claims. For example, "microcomputer" is used in some contexts to mean that microcomputer requires a memory and "microprocessor" does not. The usage herein is that these terms can also be synonymous and refer to equivalent things. The phrase "processing circuitry" or "control circuitry" comprehends ASICs (Application Specific Integrated Circuits), PAL (Programmable Array Logic), PLAs (Programmable Logic Arrays), decoders, memories, non-software based processors, or other circuitry, or digital computers including microprocessors and microcomputers of any architecture, or combinations thereof. Memory devices include SAM (Static Random Access Memory), DRAM (Dynamic Random Access Memory), pseudo-static RAM, latches, EEPROM (Electrically-Erasable Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory), registers, or any other memory device known in the art. Words of inclusion are to be interpreted as nonexhaustive in considering the scope of the invention.

67

While the presently preferred embodiments of the present invention that are disclosed in the above-identified sections are provided for the purposes of disclosure, alternative embodiments, changes and modifications in the details of construction, interconnection and arrangement of parts will readily suggest themselves to those skilled in the art after having the benefit of this disclosure. This invention is therefore not necessarily limited to the specific examples illustrated and described above. All such alternative embodiments, changes and modifications encompassed within the spirit of the invention are included.

What is claimed is:

1. A distributed telecommunications switching subsystem, comprising:

a controller operable to generate and transmit a control signal having a plurality of credit allowance values;

a first switching unit operable to receive the control signal and to transmit a first plurality of data packets in response to a first one of the credit allowance values; and

a second switching unit operable to receive the control signal and to transmit a second plurality of data packets in response to a second one of the credit allowance values.

2. The distributed telecommunications switching subsystem of claim 1, further comprising a third switching unit operable to generate a status feedback signal including a first traffic congestion level, and to transmit the status feedback cell to the second switching unit.

3. The distributed telecommunications switching subsystem of claim 2, wherein the second switching unit comprises:

a processor operable to receive the status feedback signal and add a second traffic congestion level to the status feedback signal; and

a transmitter operable to transmit the status feedback signal to the first switching unit.

4. The distributed telecommunications switching subsystem of claim 3, wherein the first switching unit comprises:

a receiver operable to receive the status feedback signal; and

a transmitter operable to transmit the status feedback signal to the controller.

5. A method for transmitting data in a telecommunications network, comprising the steps of:

generating at a controller a control signal having a plurality of credit allowance values;

receiving at a first switching unit the control signal;

receiving at the first switching unit a first plurality of customer-generated data packets;

receiving at a second switching unit a second plurality of customer-generated data packets;

transmitting the second plurality of customer-generated data packets by the second switching unit to the first switching unit;

transmitting by the first switching unit the first plurality of customer-generated data packets to a third switching unit in response to a first one of the credit allowance values of the control signal; and

transmitting the second plurality of customer-generated data packets by the first switching unit to the third switching unit.

6. The method of claim 5, further comprising the step of receiving at the second switching unit the control signal,

68

wherein the step of transmitting the second plurality of customer-generated data packets by the second switching unit is performed at selected times in response to a second one of the credit allowance values.

7. The method of claim 5, wherein the step of generating the control signal comprises the step of generating a control cell having a plurality of credit allowance words, each credit allowance word including a corresponding one of the credit allowance values.

8. The method of claim 5, wherein the step of receiving at the first switching unit the first plurality of customer-generated data packets comprises the steps of:

receiving at an input buffer of the first switching unit the first plurality of customer-generated data packets;

transmitting the first plurality of customer-generated data packets by the input buffer to a processor of the first switching unit; and

transmitting the first plurality of customer-generated data packets by the processor to an ingress queue of the first switching unit.

9. The method of claim 8, further comprising the step of performing a validity check by the processor on an address portion of each one of the first plurality of customer-generated data packets.

10. The method of claim 5, further comprising the step of retransmitting the control signal by the first switching unit to the second switching unit.

11. The method of claim 5, further comprising the steps of:

generating at the second switching unit a status feedback signal;

transmitting by the second switching unit the status feedback signal; and

receiving at the controller the status feedback signal.

12. The method of claim 11, wherein the step of generating the control signal is performed in response to the status feedback signal.

13. The method of claim 5, further comprising the steps of:

generating at the second switching unit a status feedback signal;

transmitting by the second switching unit the status feedback signal to the first switching unit;

altering by the first switching unit the status feedback signal;

transmitting by the first switching unit the status feedback signal; and

receiving at the controller the status feedback signal.

14. The method of claim 13, wherein the step of generating the status feedback signal comprises the step of generating a status feedback cell having a first status feedback word.

15. The method of claim 14, wherein the step of altering the status feedback signal comprises the step of writing into the status feedback cell a second status feedback word.

16. The method of claim 15, wherein the step of generating the control signal comprises the steps of:

calculating the first credit allowance value in response to the first and second status feedback words;

calculating the second credit allowance value in response to the first and second status feedback words; and

generating the control cell having first and second credit allowance words, the first credit allowance word including the first credit allowance value, the second credit allowance word including the second credit allowance value.

69

17. A distributed telecommunications switching subsystem, comprising:

- a controller operable to generate and transmit a control signal, the control signal having a first credit allowance word and a second credit allowance word;
- a first customer premises equipment (CPE) unit operable to generate and transmit a first plurality of data packets;
- a first switching subsystem in communication with the first CPE unit, the first switching subsystem having a first receiving system, a second receiving system, a first processor, and a first transmitting system, the first receiving system operable to receive the control signal, the second receiving system operable to receive the first plurality of data packets, the first processor operable to read the first credit allowance word of the control signal, and the first transmitting system operable to transmit selected ones of the first plurality of data packets at selected times in response to the first credit allowance word;
- a second CPE unit operable to generate and transmit a second plurality of data packets; and
- a second switching subsystem in communication with the first switching subsystem, the controller and the second CPE unit, the second switching subsystem having a third receiving system, a fourth receiving system, a fifth receiving system, a second processor, a second transmitting system, and a third transmitting system, the third receiving system operable to receive the control signal from the controller, the fourth receiving system operable to receive the second plurality of data packets, the fifth receiving system operable to receive the selected ones of the first plurality of data packets, the second processor operable to read the second credit allowance word of the control signal, the second transmitting system operable to retransmit the control signal, and the third transmitting system operable to retransmit the selected ones of the first plurality of data packets and operable to transmit selected ones of the second plurality of data packets at selected times in response to the second credit allowance word.

18. The distributed telecommunications switching subsystem of claim 17, wherein the first processor is further operable to generate a status feedback word, the status feedback word including an indication of a congestion level of the first switching subsystem and wherein the first transmitting system is operable to transmit the status feedback word to the second switching unit.

19. The distributed telecommunications switching subsystem of claim 17, wherein the first switching subsystem further comprises a scheduler in communication with the first transmitting system, the scheduler being operable to trigger transmission of the selected ones of the first plurality of data packets by the first transmitting system.

20. The distributed telecommunications switching subsystem of claim 18, wherein the second processor is operable to insert a congestion level of the second switching unit into the status feedback word.

70

21. A channel bank for a service access multiplexer, comprising:

- a first buffer operable to receive a control asynchronous transfer mode cell generated from a local processor;
- a second buffer operable to receive a bypass asynchronous transfer mode cell from a downstream channel bank;
- a third buffer operable to receive an ingress asynchronous transfer mode cell from one of a plurality of subscribers;
- a cell scheduler operable to select one of the first, second, and third buffers for upstream transport of its respective asynchronous transfer mode cell, the cell scheduler controlling upstream transport of asynchronous transfer mode cells over a pre-defined control period in response to a credit allowance received by the cell scheduler, the credit allowance determining a number of asynchronous transfer mode cells to be transported upstream by the cell scheduler during the pre-defined control period.

22. The channel bank of claim 21, wherein the pre-defined control period is a set time interval.

23. The channel bank of claim 21, wherein the pre-defined control period is a set number of asynchronous transfer mode cells to be transported upstream.

24. The channel bank of claim 21, wherein the cell scheduler is operable to give a higher priority to the first buffer, an intermediate priority to the second buffer, and a lower priority to the third buffer.

25. The channel bank of claim 21, wherein upstream transport of control asynchronous transfer mode cells from the first buffer and bypass asynchronous transfer mode cells from the second buffer do not count against the credit allowance assigned to the cell scheduler.

26. The channel bank of claim 21, wherein the cell scheduler includes a credit counter initialized to a value of the credit allowance, the credit counter operable to decrement the value of the credit allowance for each ingress asynchronous transfer mode cell transported upstream by the cell scheduler.

27. The channel bank of claim 26, wherein the credit counter is re-initialized to the value of the credit allowance upon expiration of the pre-defined control period.

28. The channel bank of claim 27, wherein the cell scheduler transports ingress asynchronous transfer mode cells upstream according to a remaining value of the credit counter from a previous pre-defined control period prior to decrementing the credit counter with the re-initialized value of the credit allowance in a current pre-defined control period.

29. The channel bank of claim 26, wherein the cell scheduler prevents ingress asynchronous transfer mode cells from being transported upstream from the third buffer when the credit counter reaches a value of zero.

* * * * *



US006611872B1

(12) **United States Patent**
McCanne

(10) **Patent No.: US 6,611,872 B1**

(45) **Date of Patent: Aug. 26, 2003**

(54) **PERFORMING MULTICAST COMMUNICATION IN COMPUTER NETWORKS BY USING OVERLAY ROUTING**

(75) **Inventor: Steven McCanne, Berkeley, CA (US)**

(73) **Assignee: FastForward Networks, Inc., San Francisco, CA (US)**

(*) **Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**

(21) **Appl. No.: 09/323,869**

(22) **Filed: Jun. 1, 1999**

Related U.S. Application Data

(60) **Provisional application No. 60/115,454, filed on Jan. 11, 1999.**

(51) **Int. Cl.⁷ G06F 15/173; H04L 12/28**

(52) **U.S. Cl. 709/238; 370/390**

(58) **Field of Search 709/238-244; 370/390**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,956,716 A	9/1999	Kenner et al.	707/10
6,003,030 A	12/1999	Kenner et al.	707/10
6,052,718 A	4/2000	Gifford	709/219
6,112,239 A	8/2000	Kenner et al.	709/224
6,415,323 B1 *	7/2002	McCanne et al.	709/225

FOREIGN PATENT DOCUMENTS

EP 0 598 969 A1 * 11/1992

OTHER PUBLICATIONS

Touch, J., *The X-Bone*, USC/Information Sciences Institute, pp. 1-3, Mar. 1997.*

Deering S. and Cheriton D., *Multicast Routing in Datagram Internetworks and Extended LANs*. ACM Transactions on Computer Systems, vol. 8, No. 2, May 1990, pp. 85-110.

Maffeis S., Bisehotberger W.R. and Matzel K.-U., *GTS: A Generic Multicast Transport Service*. In UBLAB Technical Report 94.6.1., 1994.

C. Bormann, J. Ott, H. C. Gehrcke, T. Kersch, and N. Seifert, "Mtp-2: Towards achieving the S.E.R.O. properties for multicast transport," in International Conference on Computer Communications Networks, (San Francisco, California), Sep. 1994.

Parsa M. and Garcia-Luna-Aceves J., "Scalable Internet Multicast Routing", Proceedings of ICCCN 95, Las Vegas, Nevada, Sep. 1995.

Thyagarajan, A. S. and Deering S. E., *Hierarchical Distance-Vector Multicast Routing for the MBone*, Sigcomm'95, Cambridge, Massachusetts, Aug. 1995.

(List continued on next page.)

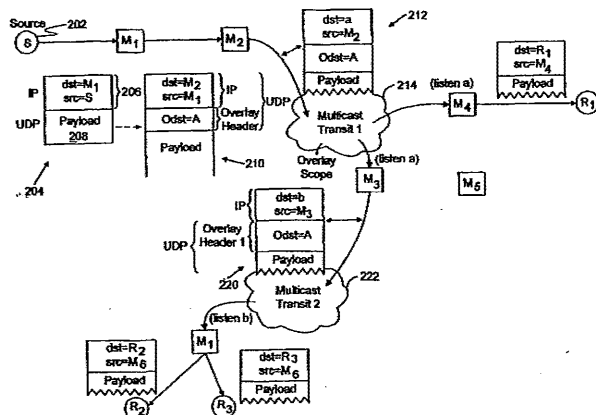
Primary Examiner—Andrew Caldwell

(74) *Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP; Philip H. Albert

(57) **ABSTRACT**

An overlay protocol and system for allowing multicast routing in the Internet to be performed at the application level. The overlay protocol uses "native" Internet multicast and multicast routing protocols to route information, according to overlay routing tables. Overlay groups are mapped to native multicast groups to exploit native multicasting in regional or local forwarding domains. Use of the overlay protocol allows overlay distribution to be handled in a more intelligent and bandwidth-managed fashion. Overlay routers are placed at each of several local area networks, Internet service provider's point of presence, enterprise, or other cohesively-managed locations. The overlay computers are configured according to bandwidth and security policies, and perform application-level multicast distribution across the otherwise disjoint multicast networks by using the overlay routing. The result is an overlay multicast network that is effectively managed according to local network management policies. Application-level control can be applied to the transferred data at the overlay routers.

7 Claims, 9 Drawing Sheets



OTHER PUBLICATIONS

- Yavatkar R., Griffioen J. and Sudan M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications", ACM Multimedia 1995, San Francisco, California, Nov. 1995.
- Amir E., McCanne S., and Zhang H., An Application Level Video Gateway In Proc. ACM Multimedia 1995, San Francisco, CA, Nov. 1995.
- Lin J.C. and Paul S., Rmtp: A reliable multicast transport protocol. Proceedings of the IEEE INFOCOM '96 pages 1-19.
- Deering S., Estrin D., Farinacci D., Jacobson V., Liu C. and Wei L. An Architecture for Wide-Area Multicast Routing. WEE/ACM Transactions on Networking, vol. 4, No. 2, Apr. 1996.
- Atwood, J. W., Catrina, O., Fenton, J., and Strayer, W. Timothy, Reliable Multicasting in the Xpress Transport Protocol Proceedings of the 21st Local Computer Networks Conference, Minneapolis, Minn., Oct. 13-16, 1996.
- Perkins C. and Crowcroft J., "Real-time audio and video transmission of IEEE GLOBECOM '96 over the internet." IEEE Communications Magazine, vol. 35, pp. 30-33, Apr. 1997.
- Sharma P., Estrin D., Floyd S. and Jacobson V., Scalable Timers for Soft State Protocols, Proceedings IEEE Infocom '97. Kobe, Japan, Apr. 1997.
- Mitra S., "Iolus: A framework for scalable secure multicasting," ACM Computer Communication Review, vol. 27, pp. 277-288, Oct. 1997. ACM SIGCOMM'97, Sep. 1997.
- Hodel H., "Policy Tree Multicast Routing: An Extension to Sparse Mode Source Tree Delivery," ACM Computer Communication Review, vol. 28, No. 2, pp. 78-97, Apr. 1998.
- Kumar S., Radoslavov P., Thaler D., Alaettinoglu C., Estrin D., and Handley M., The MASC/BGMP Architecture for Inter-domain Multicast Routing, Proc. of SIGCOMM '98, Sep. 1998, Vancouver, B.C.
- Decasper D., Dittia Z., Parulkar G, Plattner B. Router plugins: a software architecture for next generation routers Proceedings of the ACM SIGCOMM '98, Vancouver, British Columbia, Canada, Sep. 1998.
- Handley M., Crowcroft J., Bormann C., Ott J., Very large conferences on the Internet: the Internet multimedia conferencing architecture, Computer Networks (31)3 (1999) pp. 191-204.
- Holbrook H. and Cheriton D. R., IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. Computer Communication Review, a publication of ACM SIGCOMM, vol. 29, No. 4, Oct. 1999.
- Casner S., Frequently asked questions (FAQ) on the multicast backbone (MBONE). On-line documentation available from [ftp://venera.isi.edu/mbone/faq.txt](http://venera.isi.edu/mbone/faq.txt), Jan. 1993.
- Finlayson, Ross, The Multicast Attribute Framing Protocol, <http://www/live.com/mafp.txt>, Jan. 7, 1999.
- Smallcast Technical white paper Cisco Systems <http://cco.cisco.com/warp/public/cc/cisco/mkt/ios/tech/ipv/tech/hqv/wp.htm>.
- Real Broadcast Network (RBN) Splitters www.real.com/solutions/rbn/whitepaper.html.
- Farinacci D., Lin A., Speakman T., and Tweedly A., Pretty good multicast (PGM) transport protocol specification, Internet Draft, Internet Engineering Task Force, Jan. 1998.
- Farinacci D., Wei L., and Meylor J., Use of anycast clusters for inter-domain multicast routing, Internet Draft, Internet Engineering Task Force, Mar. 1998.
- Robertson K., Miller K., White M., and Tweedly A., Star-Burst multicast file transfer protocol (MFTP) specification, Internet Draft, Internet Engineering Task Force, Apr. 1998.
- Banerjee A., Faloutsos M., and Pankaj R., Designing QoS-MIC: a quality of service sensitive multicast internet protocol, Internet Draft, Internet Engineering Task Force, May 1998.
- Banerjee A., Faloutsos M., and Pankaj R., Designing QoS-MIC: a quality of service sensitive multicast internet protocol, Internet Draft, Internet Engineering Task Force, May 1998.
- Thaler D., Estrin D. Meyer D. Border Gateway Multicast Protocol (BGMP): Protocol Specification Internet Draft Nov. 1998.
- Casey L., Cunningham I. and Eros R., IP VPN Realization using MPLS Tunnels, Internet Engineering Task Force, Nov. 1998.
- Finlayson R. A More loss-Tolerant RTP Payload format for MP3 Audio. Internet Draft Jan. 1999.
- Finlayson R. IP Multicast and Firewalls. Internet Draft. May 1999.
- Blazevic L. and Le Boudec J., Distributed Core Multicast (DCM): a routing protocol for many small groups with application to mobile IP telephony, Internet Engineering Task Force, Jun. 1999.
- Hampton D., Oran D., Salama H. and Shah D., The IP Telephony Border Gateway Protocol (TBGP), Internet Engineering Task Force, Jun. 1999.
- Finlayson R. The UDP Multicast Tunneling Protocol. Internet Draft Jul. 1999.
- Malis A., Heinanen J, Armitage G, Gleeson B A Framework for IP Based Virtual Private Networks Internet Draft Aug. 1999.
- Waitzman D., Partridge C., Deering S.E., Distance Vector Multicast Routing Protocol. Request for Comments RFC 1075, Nov. 1988.
- Hanks S., Li T. Farinacci D. Traina P. Generic Routing Encapsulation Request for Comments RFC 1701 Oct. 1994.
- Farinacci D., Hanks S., Li T., Traina P. Generic Routing Encapsulation over IPv4 networks. Request for Comments RFC 1702 Nov. 1994.
- Schulzrinne H., Casner S., Frederick R., Jacobson V. RTP: A transport protocol for real-time applications. Request for Comments RFC 1889, Jan. 1996.
- Fenner W. Internet Group Management Protocol, Version 2, RFC2236 Nov. 1997.
- Ballardie A. Core Based Trees (CBT) Multicast Routing Architecture. Request for Comments, RFC 2201, Sep. 1997.
- Estrin D., Farinacci D., Helmy A., Thaler D., Deering S., Handley M., Jacobson V., Liu C, Sharma P., Wei L. Protocol Independent Multicast-Sparse Mode (PIM-SM). Request for Comments RFC 2362, Jun. 1998.
- Arango M., Dugan A., Elliott I., Huitema C. and Pickett S., Media Gateway Control (MGCP) Version 1.0 RFC 2705 Oct., 1999.

* cited by examiner

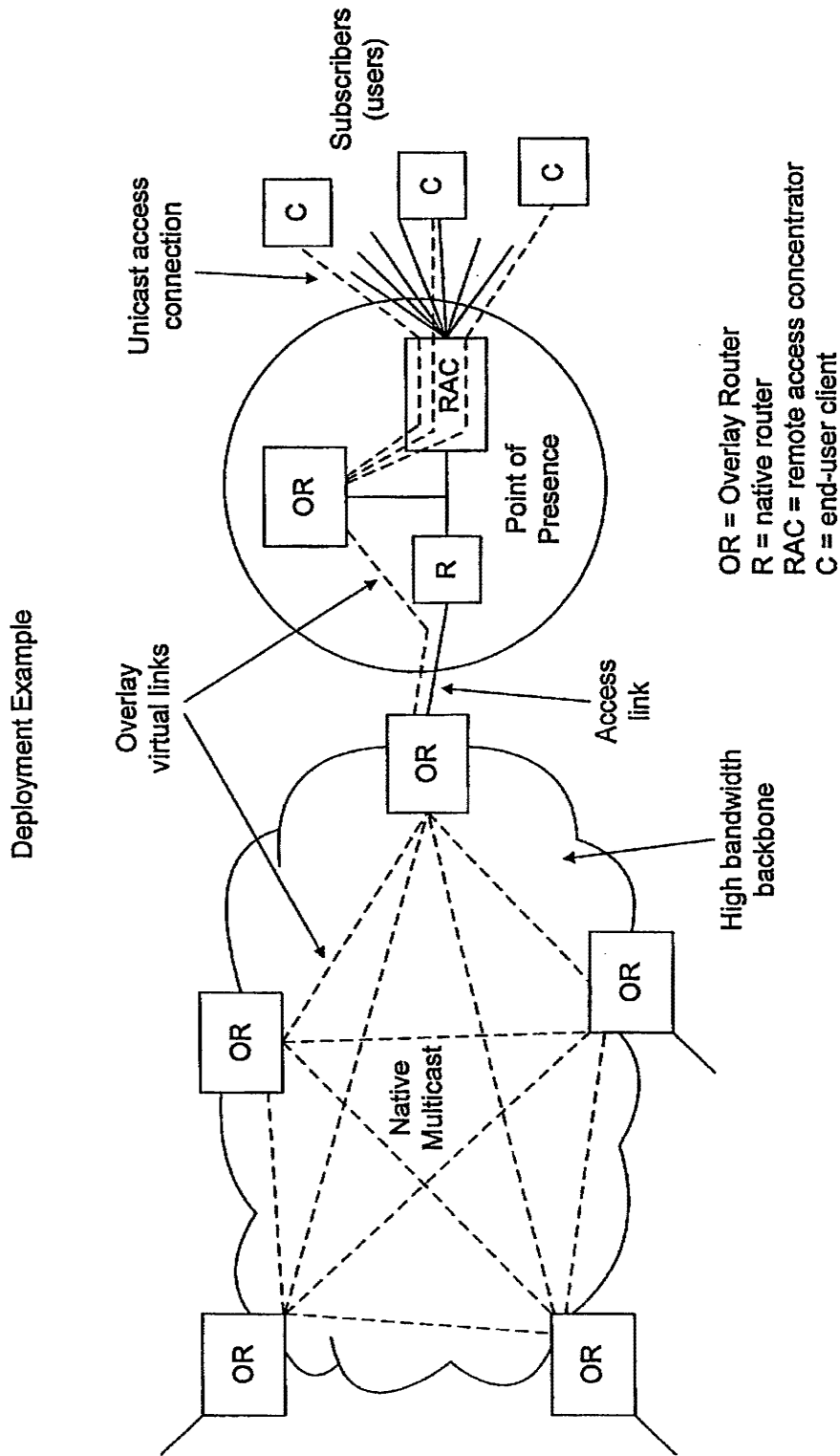


Fig. 1

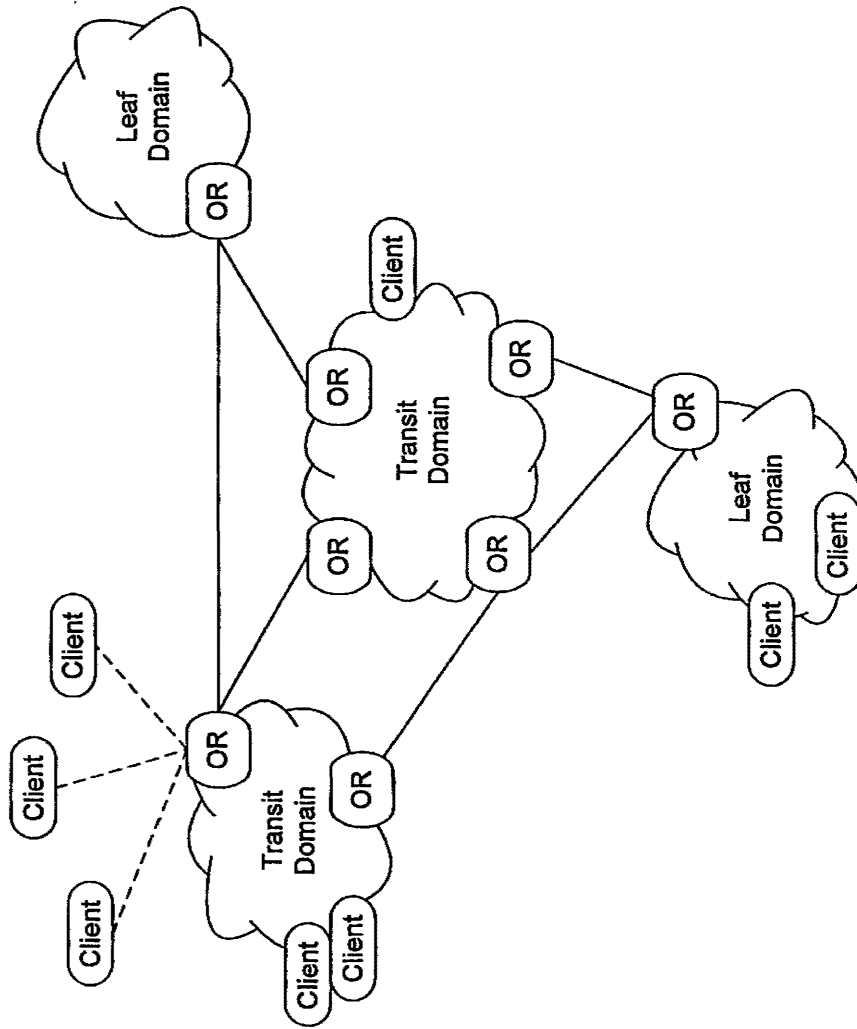
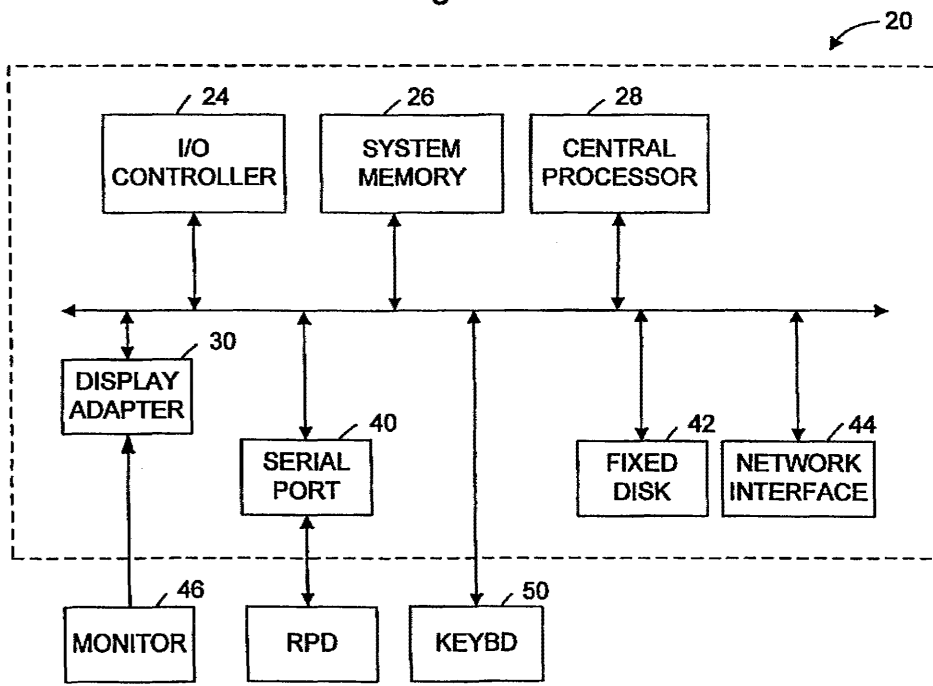
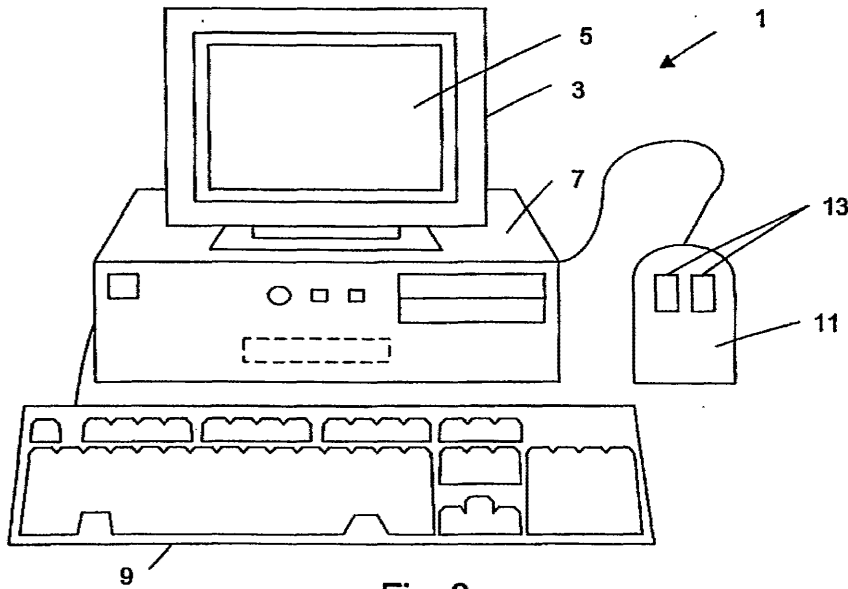


Figure 2: OMN Architecture



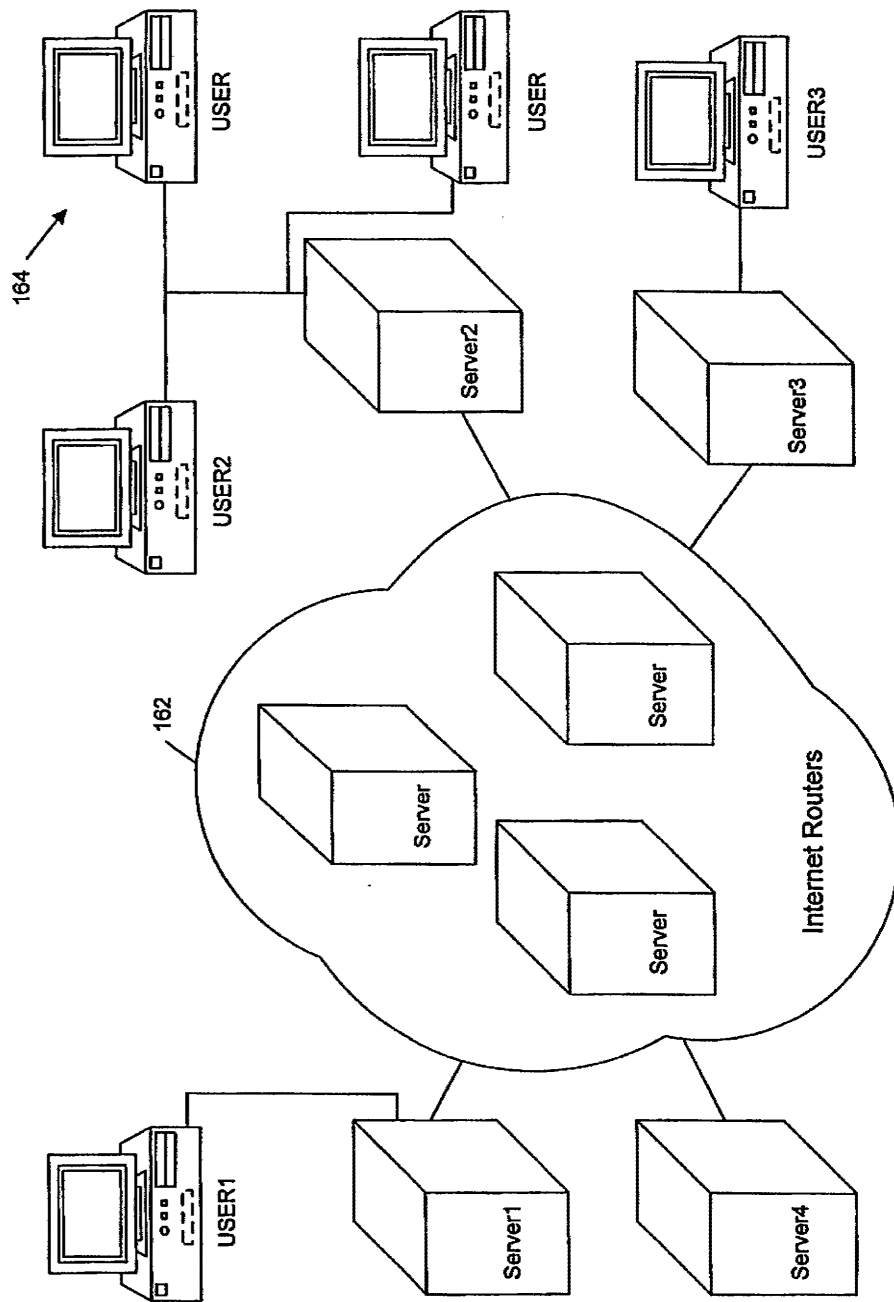


Fig. 3c

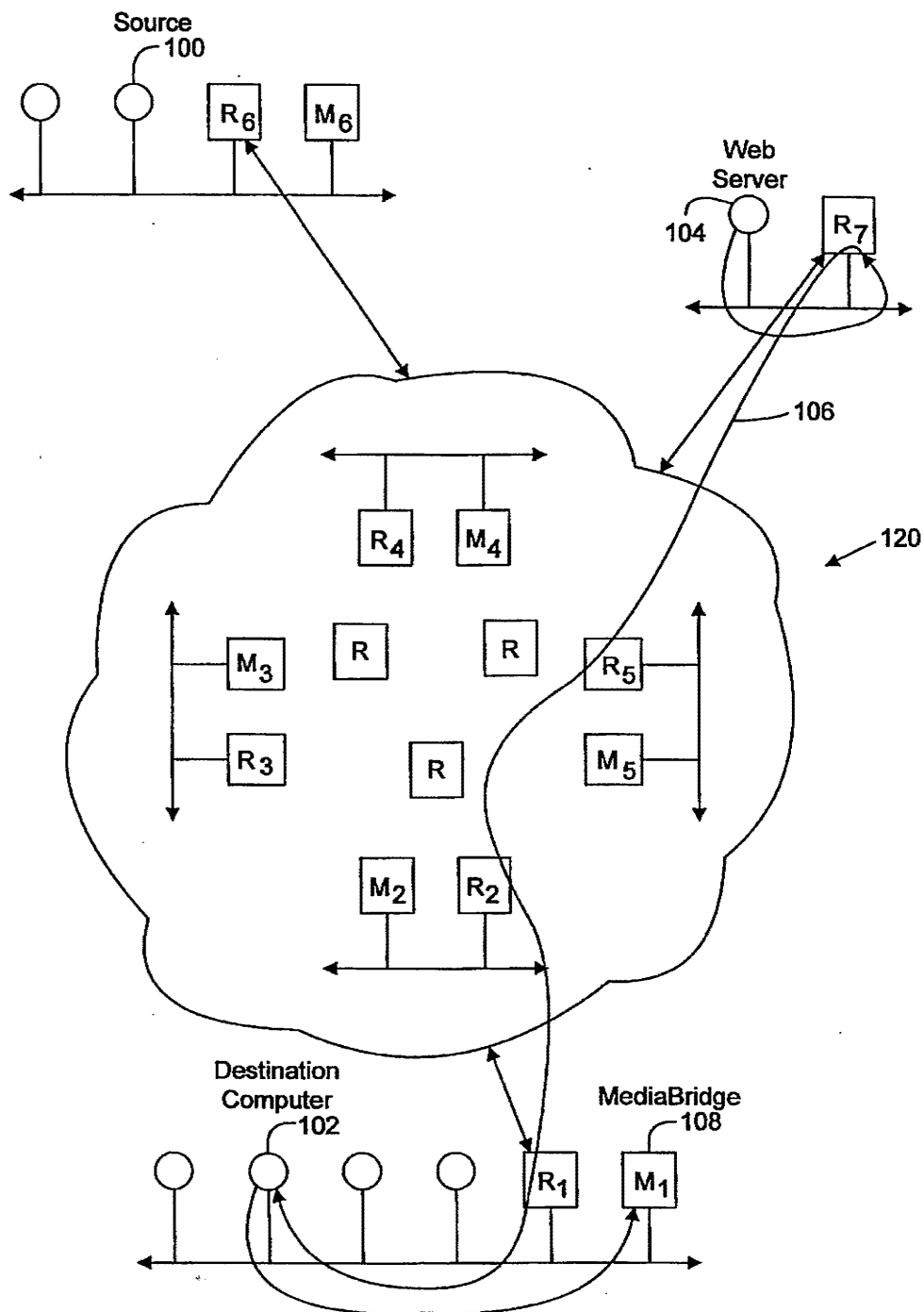


Fig. 4a

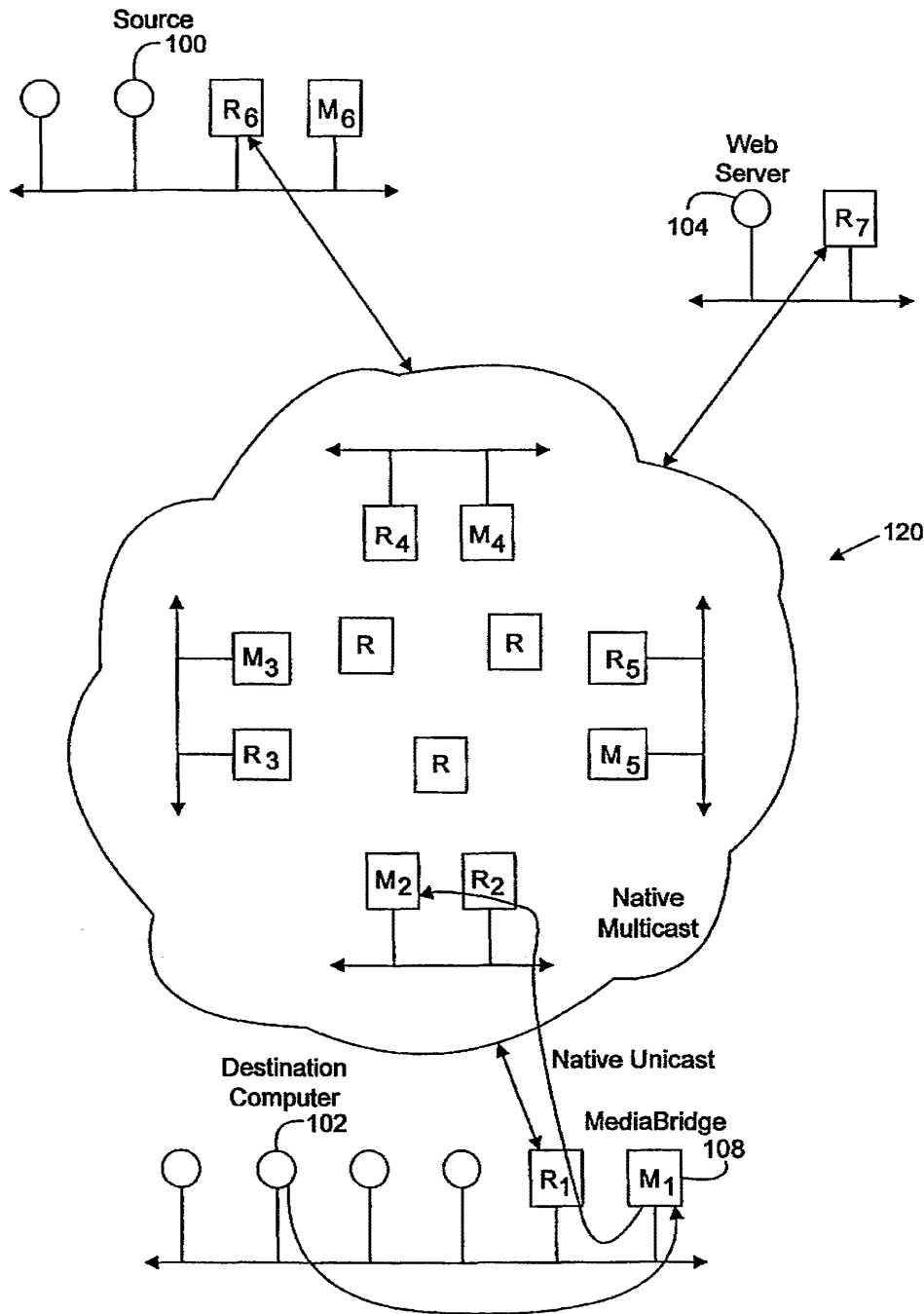


Fig. 4b

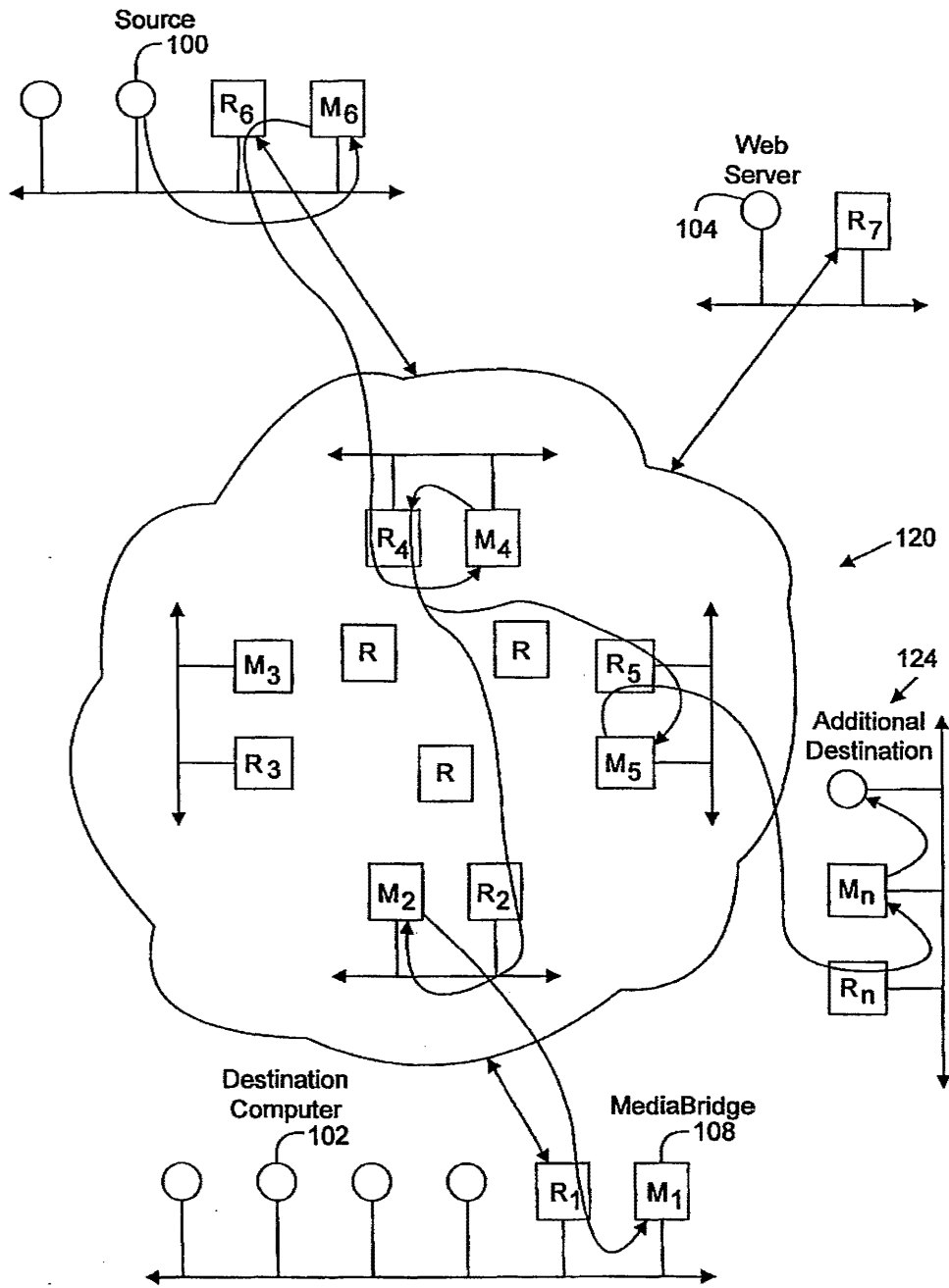


Fig. 4c

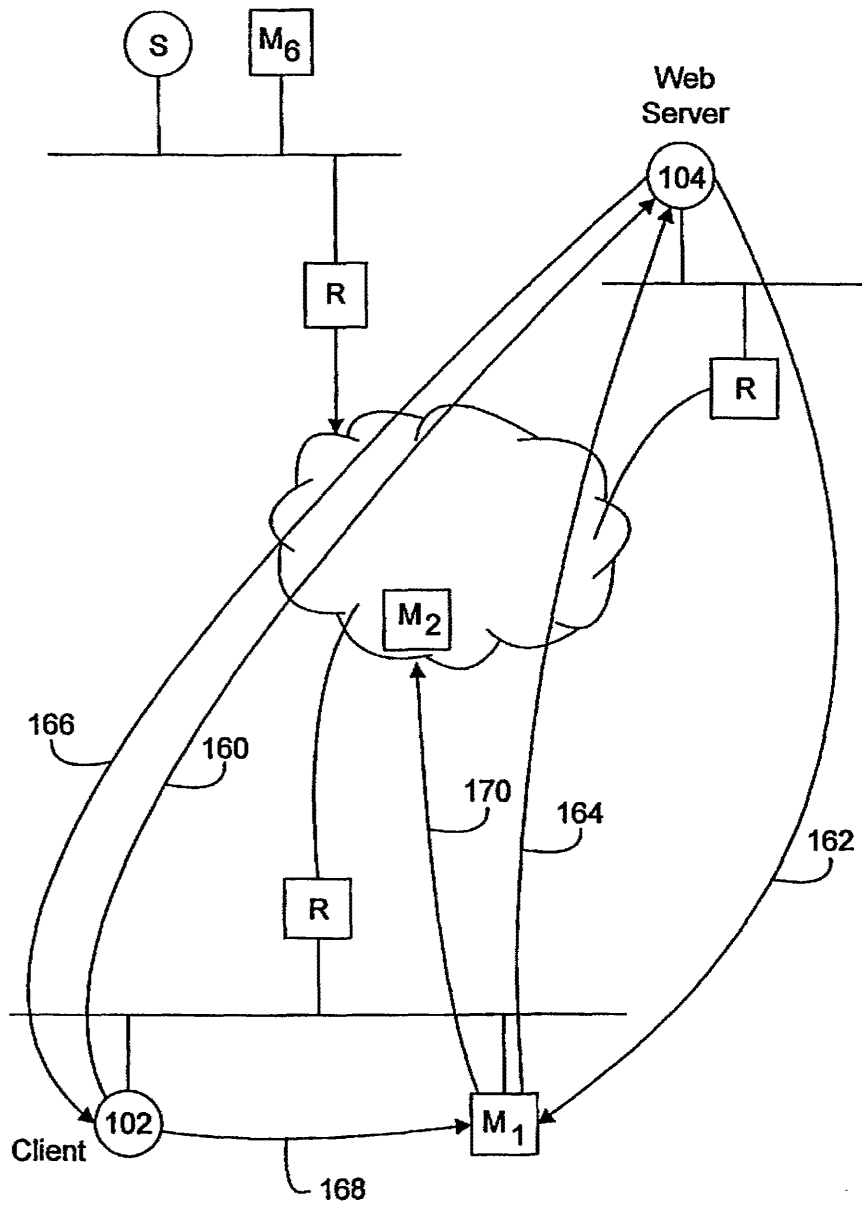


Fig. 5

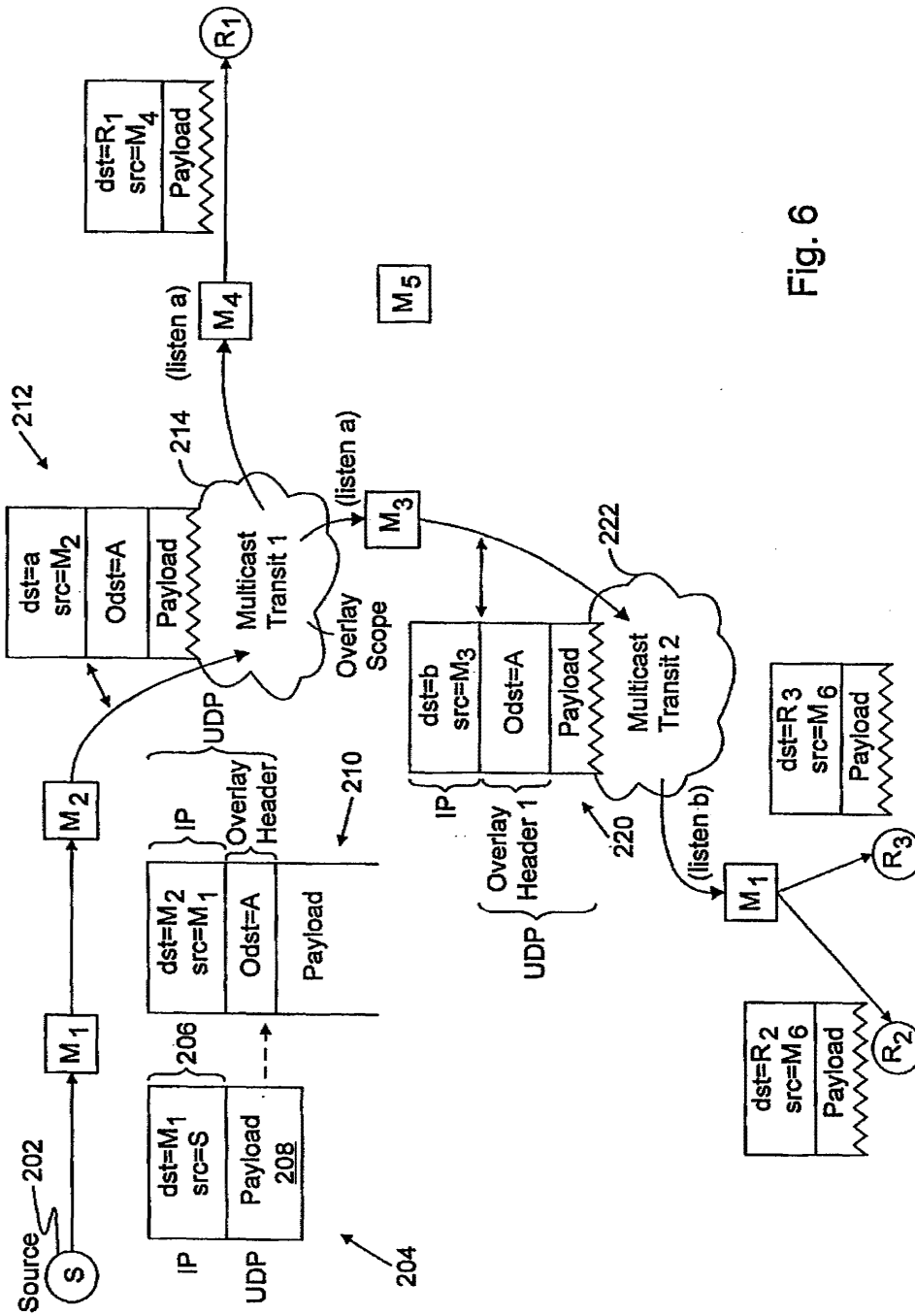


Fig. 6

**PERFORMING MULTICAST
COMMUNICATION IN COMPUTER
NETWORKS BY USING OVERLAY
ROUTING**

**CROSS-REFERENCES TO RELATED
APPLICATIONS**

This application claims priority from U.S. Provisional Application Ser. No. 60/115,454, filed Jan 11, 1999, which is incorporated herein by reference. This application is related to the following co-pending patent applications and/or provisional applications which are hereby incorporated by reference as if set forth in full in this specification: Provisional Patent application entitled "SYSTEM FOR BANDWIDTH ALLOCATION IN A COMPUTER NETWORK" filed on Jun. 1, 1999; and Provisional Patent application entitled "SYSTEM FOR MULTIPOINT INFRASTRUCTURE TRANSPORT IN A COMPUTER NETWORK" filed on Jun. 1, 1999.

BACKGROUND OF THE INVENTION

As the Internet gains in popularity it is desirable to allow broadcasts of live media, such as a television program or radio program, over the Internet. However, a problem with such "streaming media" broadcasts is that they require very high data transfer rates across many servers, routers and local area networks that form the Internet. Because of this, high-quality, scalable broadcasts, or "multicasts," of streaming media information to massive numbers of end-users at once over the Internet have not been achieved to date.

Examples of attempts to design and deploy multicast systems for the Internet include systems by RealNetworks and Broadcast.com. RealNetworks has built an Internet broadcast infrastructure called the Real Broadcast Network (RBN) while Broadcast.com has partnered with Internet Service Providers (ISPs) and content providers to build a broadcast system based on native Internet multicast routing. A typical streaming broadcast consists of a server that unicasts a User Datagram Protocol (UDP) flow to each requesting client. Bandwidth is managed very crudely by simply limiting the number of simultaneous active flows via some simple configuration hook at the server. While this approach works for today's demands, it wastes network bandwidth and cannot scale to very large audiences anticipated in the future.

Although some existing products (like NetShow and Cisco's IP/TV) support multicast, Internet Service Providers (ISPs) and enterprise network managers have been slow to adopt multicast because it is difficult to configure, manage, and debug. For some companies, these deployment problems are crucial barriers because they view multicast as critical for their long-term viability and feel that failure to successfully integrate multicast could compromise their mission.

A successful Internet broadcast system depends on its ability to broadcast audio and video programming to a large number of simultaneous users. Two approaches for broadcasting streaming media are replicated unicast (one user per stream) and multicasting (multiple users per stream).

While unicast delivery has enjoyed tremendous success as the fundamental building block of the Internet, multicast has proven far more complex and many technical barriers remain that prevent multicast from being deployed across the wide area. Despite a decade of research and development, interdomain multicast routing has yet to be successfully realized and there are many reasons to believe

that multicast, in its present form, may never be universally deployed throughout the Internet. In this case, applications that assume ubiquitous multicast connectivity to attain scalable performance will never become feasible. On the other hand, multicast, when restricted to a singly administered network domain, has been much easier to configure and manage, and for this reason, has been a resounding success in isolated deployments. That is, it's easy to build an isolated multicast "cloud" as long as it doesn't span multiple administrative domains and involve highly heterogeneous equipment with different multicast implementations.

While uniform and homogeneous multicast clouds can effectively carry multicast traffic that is isolated to LANs or autonomous corporate networks, a wide range of compelling applications, such as streaming media broadcasts, are potentially enabled by interconnecting the isolated multicast clouds into a very large-scale distribution network. However, attempts to use wide area multicast routing protocols to solve this problem have failed. Another problem with interconnecting multicast clouds has been lack of control over the multicast traffic from domain to domain. This implicates not only bandwidth considerations, but security issues as well.

Thus it is desirable to complement and enhance the vast array of existing servers and end-clients with a state-of-the-art system that provides an improved network infrastructure for achieving multicasting of information. Such a system should enhance existing digital audio/video/media applications and enable them to work more effectively at large scale and across heterogeneous environments. The system should provide flexible bandwidth management and diagnostic tools to network managers such as by providing localized control over traffic and content of multicast data. The system should make use of existing, widely deployed communication protocols and procedures to achieve efficient transfer of information.

SUMMARY OF THE INVENTION

The present invention is to be embodied in a commercial product by FastForward Networks, called "MediaBridge." Each MediaBridge is a process that is executed on a computer, or other processor or processing device, connected to a network. Multiple MediaBridges transfer data using an "overlay" network. In a preferred Internet embodiment, the overlay protocol uses "native" Internet protocols to route information, according to overlay routing tables, between otherwise disjoint and isolated multicast clouds. This allows the overlay distribution to be handled in a more intelligent and bandwidth-managed fashion. For example, MediaBridges are placed at each of several local area networks (LANs), ISP "point of presence" (POP), enterprise, or other cohesively-managed locations. The MediaBridges are configured according to bandwidth and security policies, and perform application-level multicast distribution across the Network Access Points (NAPs) using overlay routing. The result is an overlay multicast network that is effectively managed according to traffic policies defined locally at each NAP.

The present invention allows application-level control to be applied to the transferred data. For example, if a confluence of high-bandwidth video flows arrives at a choke point in the network (where the choke point is either a physical bandwidth limit or an administratively configured bandwidth constraint), the MediaBridge intelligently filters and/or transforms flows so that they fit onto the outgoing link. The transformations can, for example, reduce the frame rate

or resolution uniformly as opposed to simply dropping packets at the network layer (without accounting for media semantics). The invention exploits application-level activity to control adaptation. For example, in a videoconference, cues from the audio channel, or from the dispositions of the user interfaces at the clients, can be used to decide to dedicate more of the traffic class' bandwidth allocation to the current speaker.

An end-user client application can attach to the overlay network using either unicast or multicast communication between it and a MediaBridge on the overlay. Thus, a web page can have a simple "point and click" hyperlink to initiate reception of a multicast audio/video production where a channel ID is embedded in the Uniform in Resource Locator (URL). Or a user can send a channel ID, or other identification, to a MediaBridge to subscribe to a program multicast.

In one embodiment of the invention an overlay routing processor for transferring information over a computer network is disclosed. The computer network has a native routing protocol. The overlay routing processor includes instructions for associating computers on the network with a given overlay group; instructions for determining whether received information is associated with the given overlay group; and instructions for routing the received information to the computers associated with the given overlay group by using the native routing protocol.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows overlay router arrangements;

FIG. 2 illustrates the Overlay Multicast Network Architecture;

FIG. 3A shows a computer suitable for use with the present invention;

FIG. 3B shows subsystems in the computer of FIG. 3A;

FIG. 3C illustrates a network configuration;

FIG. 4A shows a unicast-initiated overlay routing step;

FIG. 4B illustrates a second step in overlay routing;

FIG. 4C illustrates a third step in overlay routing;

FIG. 5 illustrates a second approach to unicast-initiated overlay routing; and

FIG. 6 illustrates details of header and address processing in the present invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

The present invention implements "overlay" multicasting. So-called because some of the routing processing by MediaBridge's uses a routing scheme that is independent of, in addition to, and at a higher level than the prior art "native" scheme. With the approach of the present invention, any of the current multicasting techniques, such as DVMRP, PIM, CBT, etc. are referred to as "native" multicasting," or "native protocols."

The invention uses native multicast only as a forwarding optimization where it is locally viable—typically within medium-scale, singly-administered, homogeneous network domains. In this model, the network consists of a set of isolated native multicast clouds upon which a virtual network of application-level routing agents called "overlay routers". The overlay routers (i.e., the MediaBridge computers) implement multicast routing protocol that makes use of sophisticated application-level knowledge and management infrastructure. Unicast clients can connect directly

to overlay routers via unicast so that regions of the network that do not provide native multicast support can be reached. Unlike existing multicast technologies, this allows clients to connect to overlay routers using unicast UDP or TCP through a redirection and location service. This allows clients that are not directly attached to a multicast capable network to communicate and receive transmissions from the overlay network.

Moving wide-area multicast routing out of the network layer and up to the application layer, affords a number of advantages:

Simplicity. The overall multicast routing problem is simplified because it is decomposed into two separable and individually tractable sub-problems: (1) the configuration of native multicast routers in a singly administered network domain and (2) the configuration of overlay routers to interconnect the disparate regional networks and tightly manage the bandwidth consumed by multicast across these domains.

Rich Management. Because routing is carried out at the application layer, we can implement well-defined policies that reflect application priorities and provide high-level services such as billing. Routers are no longer hidden by a thick protocol layer that hides application knowledge. Instead, the overlay routers can be configured more like servers and bandwidth managed and apportioned intelligently across different application classes.

Efficient Rendezvous. Application-level knowledge vastly simplifies the problem of rendezvous. Because different applications naturally have different models for how the distributed components might interact or at what scale and directionality they interact, the rendezvous mechanism can be optimized by exploiting application requirements. For example, a streaming video server might best be contacted by querying the content provider's ordering service or by contacting the corporate headquarters' web page. Alternatively, a videoconference might best be initiated by contacting an H323 Multi-point control unit (MCU) that is near the physical conference room of the hosting site.

The invention is next discussed by first presenting the general architecture of the overlay multicast network approach. Next, detailed examples of transaction steps using the architecture are described.

THE OVERLAY MULTICAST NETWORK ARCHITECTURE

1. Introduction

This section describes the overlay multicast network (OMN) architecture of the present invention.

The OMN architecture utilizes a two-level addressing strategy, where overlay addresses are carried in an additional overlay header (which appears after the UDP header), but before the user's UDP payload, and native multicast addresses are computed from overlay addresses using a novel hashing scheme that exploits multicast address scopes. To properly route traffic across the overlay network, overlay routers implement a multicast routing protocol that is, in some ways, analagous to BGMP and BGP. In addition, unicast clients can connect directly to overlay routers via UDP so that regions of the network that do not provide native multicast support can be reached. The overlay routers operate at the application-level. This allows the overlay routers to be easily, extended with application-level knowl-

edge to carry out semantically-aware transformations conditioned on bandwidth constraints specified by external policies.

The OMN architecture includes a number of core elements:

- a forwarding and routing framework for computing multicast distribution tree across the virtual overlay network,
- a multipoint reliable transport protocol for disseminating data reliably into and across the overlay network,
- a plugin framework for extending overlay routers with new application-level knowledge,
- a bandwidth scheduling framework for scheduling traffic classes according to hierarchical link-sharing policies,
- a method for communicating between plugins and the bandwidth management subsystem to effect application-level adaptation from within the network, and
- a referral directory service that redirects end hosts to the closest overlay router.

This application is principally concerned with the routing components of the OMN architecture and the relationship among the different subsystems. Other related applications include those referenced at the beginning of this specification.

2. The Network Model

The network model assumed by an overlay network is a collection of isolated (but possibly overlapping) regions of native multicast connectivity. Overlay routers are deployed across this arrangement of multicast clouds and peer with each other, either via unicast or multicast UDP/IP to form a network of application-aware multicast forwarding agents. End hosts inject traffic into the overlay network using either native multicast across a "leaf scope" or using unicast communication directly to a nearby overlay router.

Even though the OMN framework operates at the application layer, overlay routers must compute what amounts to network-level routes to determine how to flood multicast flows across and throughout the appropriate region of the overlay network. Thus, in the OMN architecture routing occurs at two layers, the network layer and the application layer. Because routing is carried out the application layer, application-level knowledge can be integrated into the forwarding process to transform packet flows at points of administrative discontinuity.

In this two-layer routing model, the network (IP) source and destination addresses are rewritten on each overlay router hop, which means that certain structure and state (like address allocations and multicast spanning trees) need not be globally consistent across multicast domains. Note that this allows overlay routing without requiring all routers in the network to be upgraded to recognize and forward a new packet type. No change to the existing routing infrastructure is needed because of the two-layer addressing scheme. That is, existing multicast routers can remain intact while new overlay routers are installed at the borders of administrative boundaries, or domains. We thus exploit existing native multicast routing technology within administrative domains and across transit domains when and where available.

2.1 The Overlay Multicast Service Model

In contrast to native IP Multicast, the overlay multicast service model transforms packets as necessary in order to forward application-level flows in a bandwidth-managed

fashion. In this model, an application may inject a flow into the network without concern that it will congest the network since the overlay routers will thin the stream as necessary in choke points in the network and ensure that all policy-defined bandwidth constraints are adhered to. In addition, sources must explicitly signal to the network their intention to send and optionally indicate type information describing their traffic. Administrative policies can be configured into the infrastructure. These policies can permit or restrict sources from sending based on rich, application-level policies.

To maximize the congruence between the OMN architecture and the existent IP Multicast service interface, hosts use the standard IP Multicast interface to inject data packets into and receive packets from an OMN. In one embodiment of the invention, overlay multicast senders (or proxies for the sender) explicitly signal to the network their intention to transmit. This is unlike IP multicast, where hosts may simply send packets addressed to a Class D multicast group without any explicit signaling. As part of this dialogue, the sender describes the channel that it intends to use (e.g., UDP multicast, UDP unicast, or TCP), and, once negotiated, overlay-enabled multicast packets may be sent into the network. This sender setup process may fail if the source does not have administrative permission to send. Thus, OMN sources can be tightly controlled in contrast to normal IP multicast, which provides no control over senders.

To invoke application-level processing and management capabilities within the OMN network infrastructure, an OMN sender (or proxy thereof) may signal application knowledge into the network using a multipoint data dissemination framework. In the preferred embodiment, this framework uses a protocol known as Multipoint Infrastructure Transport (MINT) Protocol. MINT provides a group-oriented, reliable delivery mechanism between a nodes in the OMN and is described in detail in a co-pending patent application referenced at the beginning of this specification.

Using MINT, senders can attach named values to an overlay multicast group which is published into and across the overlay network, allowing other group members as well as network entities to query this "database" of state. Each tuple in the database, called a "mint", is identified by its owner (the OMN sender) and name (and implicitly the group). The tuples are disseminated reliably to all parts of the network with active participants. Note that given tuples need to flow only to overlay routers that fall along a path from the source to the set of active receivers for that group. An end host may query the OMN subsystem to discover and/or enumerate all known owners and all known keys published by each owner. In turn, the values can be queried by name/owner, and the application can be asynchronously notified when the owner modifies the value.

Certain mints are reserved for system specific functions that, for instance, map an overlay group to an application type or describe the attributes of an overlay group so that it can be mapped into locally defined traffic classes in different parts of the network. For flows that require application-level processing and/or traffic management, a special "setup mint" provides the requisite information and must precede the transmission of data. Packets are dropped by the overlay network if the setup mint is not present, including the time during which the setup mint is in transit.

2.2 Virtual Interfaces

A fundamental communication construct in overlay multicasting is a path abstraction called a "virtual link", which joins together an overlay router with other overlay routers

and with end hosts. The (virtual) attachment abstraction of a link to an overlay router is called a virtual interface or VIF. There are three primary classes of VIF: a transit VIF (TVIF) interconnects two or more overlay routers in a peering relationship, a leaf VIF (LVIF) interconnects end hosts with native multicast connectivity to the overlay network, and a unicast bank VIF (UVIF) interconnects end hosts without multicast access via unicast to a "nearby" overlay router.

Applications send and receive OMN packets through an overlay router that manages the LVIF. The overlay router transforms each native multicast packet into an overlay packet by encapsulating it in an overlay header, which is situated between the UDP header and application payload. This overlay header includes the destination overlay group, which consists of a 32-bit rendezvous point (RP) identifier and a 32-bit channel number. When a packet reaches its destination LVIF, the last-hop overlay router strips off the overlay header and forwards the packet to the end host (or set of end hosts) using unicast (or multicast) UDP.

Each transit VIF represents a link in the overlay network topology and overlay routers forward packets to each other over these virtual paths. A collection of overlay routers can peer with one another as a group over a "multicast transit VIF". Overlay routers can peer with each other directly in a "point-to-point" configuration using a "unicast transit VIF". In addition, end-hosts inject/receive packets from the overlay network by either (1) transmitting packets into a leaf VIF using native multicast or (2) transmitting packets directly to an overlay router using unicast UDP. For the latter case, the address of a nearby overlay router can be queried using a directory service.

An example of overlay routing is shown in FIG. 1. FIG. 1 shows overlay routers (ORs) arranged to handle traffic in a transit domain using native multicast, and in a point-of-presence system between a native router (R) and remote access concentrator (RAC). Many such arrangements of overlay routers are possible. The number, placement and physical connection of overlay routers is a design tradeoff with respect to desired efficiency, cost and features as is practicable.

2.2.1 Overlay Scope

When an overlay router sends an overlay packet out a VIF, it must determine the network-layer address of the "next hop". If the VIF is composed of only a single peer, then the address is simply the unicast address of that peer. But, for a collection of peers, the overlay router must map the overlay group into a native multicast group so that the peers can receive the traffic efficiently using native multicast. To provide controls over transit traffic containment, this address is chosen from a range of administratively scoped addresses, which are configured into the overlay router when the peering relationship is established. This scope is referred to as an overlay scope in the OMN architecture.

The native IP multicast service supports two type of "scoping" abstractions: hop-based scope and administrative scope. In hop-based scope, the time-to-live (TTL) field in the IP packet header constrains the distribution of a packet. Because the TTL limits the number of forwarding hops that a packet can sustain before being dropped, the source host can constrain the reach of the packet by setting the TTL field to an appropriate value. In administrative scope, routing boundaries are configured borders between scopes (e.g., between separate organizations). A routing boundary is represented by a range of multicast addresses, i.e., an administrative boundary is imposed by preventing multicast packets that fall within the administrative address range to be blocked at that boundary point. A special block of

multicast addresses is reserved for administrative scope (239.0.0.0 to 239.255.255.255) and since administratively scoped traffic does not flow across boundaries, scoped addresses need not be unique across organizational boundaries.

Associated with each VIF is a rule by which overlay multicast addresses are mapped onto the overlay scope, or range of native multicast addresses. An overlay scope is expressed as an IP4 Class D address and a prefix length, e.g., 249.2.16 represents the block of addresses from 249.2.0.0 to 249.2.255.255. An algorithm to deterministically map an overlay multicast address into an address in an arbitrary overlay scope can be easily constructed using well-known hashing techniques.

2.2.2 Leaf VIFs

Multicast traffic enters and leaves the overlay network through a special VIF called a "leaf VIF" (LVIF) (so called because these VIFs are situated at the leaves of the multi-point overlay distribution tree). FIG. 2 is an illustration of overlay routers arranged in leaf and transit domains. To contain and control traffic within the LVIF, a TTL-based multicast scope is imposed at the desired boundary of the leaf. That is, to create an LVIF, a network administrator determines which subnets in the network should comprise that LVIF, and in turn, configures each router attached to those subnets with an appropriate TTL threshold. In turn, applications and overlay routers inject traffic into the LVIF with a TTL less than the threshold thereby containing the reach of raw multicast traffic as desired.

Unfortunately, TTL scopes can lead to inefficient use of bandwidth because TTL-scoped traffic often cannot be "pruned" off subnets that have no interested receivers. To remedy this, administrative scope boundaries can be placed precisely along the border of the leaf scope in question. If applications then use addresses from these ranges, the traffic is consequently scoped and pruned off subnets (within the leaf) for which there are no receivers. However, because the overlay network effectively bridges spatially distinct multicast sub-regions, the usual locality implied by administrative scopes no longer applies. Thus, such scopes must be carefully used and set aside for use only for the overlay network (i.e., not relied upon for other uses of scoped traffic since the overlay network deliberately leaks this traffic outside the region). Another problem with administrative scopes is that different sites might choose different address ranges for scopes. Thus, the overlay multicast architecture reserves a special range of administratively scoped addresses to be used exclusively for the purpose of imposing boundaries on leaf VIFs.

Given that the LVIF scoping abstraction is in place to constrain the reach of data traffic, end hosts and overlay routers still must exchange control traffic in order to determine which traffic to forward into and out of the LVIF. That is, the overlay router at the edge of the domain must somehow determine the set of multicast groups that it must listen to in order to receive traffic from active senders. Likewise, it must determine whether receivers are present for any overlay group in question so that it can join the corresponding group across the overlay network (see the Section 4 below) and forward the consequent traffic from sources in other remote LVIFs into the local LVIF. To solve these problems, end systems and overlay routers utilize control protocols that run over well-known multicast groups and TCP to carry out the necessary state exchanges. Receiver overlay group membership is reflected to overlay routers through a protocol called the Domain-wide Group Membership Protocol (DGMP). The presence of senders for

a given overlay group is signaled though the Overlay Sender Setup Protocol (SSP). In turn, senders use MINT to further attach attributes to their flows to effect application-level processing and bandwidth management of their traffic within and across the overlay network.

To enhance fault tolerance and improve routing performance, multiple overlay routers may be attached to a single LVIF. When multiple overlay routers are incident to an LVIF, they intercommunicate with each other to elect a designated router (DR) for the LVIF. The remaining overlay routers are demoted to subordinate status. As such, only the DR injects or extracts traffic from the LVIF, while the subordinate routers act as backups in case the DR goes down. In addition, the subordinate routers may peer with the DR using the normal peering relationships (see below), which improves the routing performance by potentially reducing the path lengths from a given LVIF that may be connected to multiple external networks.

In short, to receive or send packets into an OMN Network, end hosts explicitly register their intention to do so by using a combination of protocols that run across the LVIF.

LVIF Receivers

The receiver group membership protocols are relatively straightforward compared to the sender setup process. Unlike senders, which must describe their flow attributes, receivers simply announce their interest for a particular group to the overlay routers in the LVIF using DGMP, which is a standard protocol based on the Interdomain Multicast Routing (IDMR) Working Group's protocol for "Domain Wide Multicast Group Membership Reports". Like this related work, DGMP resembles the Internet Group Management Protocol (IGMP), but rather than run on isolated LANs, it runs across the LVIF. In this scheme, one or more overlay routers are attached to an LVIF and exchange messages over a well-known multicast group (the "DGMP channel"). A distributed election algorithm chooses one overlay router to be the designated router. This router, in turn, solicits membership reports by multicasting a "query" packet on the DGMP channel. All end hosts then announce their interest in all groups they wish to receive by multicasting "report" packets on the same DGMP channel. DGMP, unlike IGMP, does not suppress duplicate report packets, which means that the overlay router and/or monitoring systems running in the LVIF can learn of all active multicast receivers (e.g., for accounting or diagnostics). To avoid traffic transients induced by synchronized feedback, end hosts wait a random amount of time before generating their report packet (such that report traffic is spread adequately across the reporting interval). Since the overlay router controls the query rate, control bandwidth overhead can be easily traded off for membership accuracy through adjustments made only to the infrastructure (i.e., the overlay router).

To support unmodified multicast clients, a third-party end system may act as a proxy for another host. That is, an IP host may respond to DGMP queries on behalf of another. With this proxy capability, an agent can be built that passively monitors IGMP messages on a LAN, converts the Class D multicast addresses carried in the IGMP packets to overlay groups, and relays this overlay group information onto the DGMP channel. If this agent is placed on each LAN within an LVIF, then no changes are needed to support the receive path of conventional, unmodified multicast applications. That is, the agent notices whenever a receiver in its attached LAN joins a multicast group and relays the appropriate overlay group membership messages on the LVIF-

wide DGMP channel. Note that under this scheme, when a proxied application exits, the IGMP reports cease, which automatically prevents further generation of DGMP messages.

Once the DR learns that receivers for a given group exist in its incident LVIF, it informs the routing subsystem which causes packets addressed to that group to be delivered from anywhere in the overlay network. When a packet addressed to that group arrives at the DR, the packet is forwarded onto the LVIF using UDP. At this point, the overlay header is removed and the packet is addressed to the UDP port indicated in said header.

Unicast Receivers

In many cases, multicast connectivity will not reach everywhere and many clients may want to connect to the overlay network without the aid of multicast. In this case, a receiver interacts directly with an overlay router with unicast communication. The receiver consults a well-known directory service to provide the IP address of an overlay router. In turn, it sends control information indicating the overlay group desired and the transport connection parameters that the overlay router should use to forward that group's traffic (e.g., UDP or TCP port numbers). In addition, the client can provide the overlay router with additional application-level information that will allow it to optimize the delivery of traffic to that destination. For example, the client's Internet access bandwidth could be included so that the overlay router can deliver a version of the overlay multicast traffic signal that will adhere to this bandwidth constraint and thus not congest the network.

Clients do not have to contact the directory service directly. For example, existing clients can be made to join the overlay group as a side effect of clicking on a web link (as discussed below).

LVIF Senders

In IP Multicast, senders simply transmit packets to a group address without invoking any sort of control protocol like IGMP. In turn, routers directly attached to the local subnet simply detect these multicast packets and forward and route them accordingly. However, this is not easily done at the application layer since an overlay router must explicitly join the native group in question in order to receive packets. Thus, the OMN architecture requires an analog protocol to DGMP for signaling the presence of sources within the LVIF to the attached overlay routers so that they can join and leave native groups as required. Additionally, the application-aware nature of the OMN infrastructure requires that source describe the characteristics and behavior of their flows. Since this state needs to be signaled reliably into the network, an end host requires a reliable transport protocol between it and its nearby overlay router to transmit this state. Rather than implement one protocol, modeled say after DGMP, for signaling the presence of senders and another that provides a reliable delivery model for transferring flow state to the overlay router, these two functions are merged into a single protocol, the Overlay Sender Setup Protocol (SSP), which in turn leverages TCP to provide reliability. Using SSP, a sender (or proxy thereof) informs the DR of the overlay group and UDP port that it uses to carry its traffic. This allows the DR to join the group in question and relay said traffic from the indicated UDP multicast group and port into the overlay network. To effect the exchange of control state, the sender establishes an SSP connection with the DR, which is determined or discovered with a or resource discovery protocol.

SSP also provides the interface for senders to publish data into the overlay network via MINT. Mints are injected into the network using a simple protocol layered on top of the SSP channel. That is, SSP provides the conduit both for senders (or proxies) to signal their intention to send to a given overlay group and for senders to publish reliably disseminated mints into the overlay network. To support unmodified, native multicast clients, a third party agent (which we call a shim) may perform the SSP and MINT signaling functions on behalf of the unmodified client or server.

Once a setup binding is signaled into the network via SSP, the state is refreshed to persist; otherwise, the DR assumes the sender is no longer present and tears down the corresponding state. This can occur directly within SSP via explicit refresh messages, or it can be refreshed indirectly as a side effect of the sender generating data packets. That is, if the sender is continually active, there is no need to maintain the SSP connection (e.g., a streaming video server that plays out a live broadcast). Otherwise, if the sender is "bursty" (i.e., alternates between active and idle periods), the sender state must be maintained by leaving the SSP connection in place (and relying upon SSP keepalives to maintain the connection). One disadvantage of the data-driven approach, however, is that if the DR crashes, the binding must be re-established when the DR resumes or when another DR takes over.

Before the network will forward a traffic for a particular overlay group, some source in the network must publish a special, system-reserved flow descriptor mint for that group, which describes the media type of the flow and provides descriptive information that allows overlay routers to map a flow onto a locally-defined traffic class. This, in turn, allows overlay routers to perform application-specific processing and traffic management. If the flow does not need to be explicit managed or processed by the OMN infrastructure, then a flow descriptor indicated such (i.e., a best effort descriptor) must still be published into the network. Packets are dropped by the OMN until the flow descriptor is disseminated properly.

It is an error for multiple flow descriptors to be published into the network for a single overlay group from different sources. If this occurs, conflict resolution heuristics are invoked, but the results are undefined. The error condition is detected and propagated to the overlay network management facilities to be fed back to the user or users causing the problem and/or to a network operator.

Unicast Senders

In many cases, multicast connectivity will not reach everywhere and many clients may want to connect to the overlay network without the aid of multicast. In this case, a sender interacts directly with an overlay router with unicast communication to transmit traffic into the overlay multicast network. The sender consults a well-known directory service to provide the IP address of an overlay router at the edge of the OMN network. In another configuration, the sender can be statically configured with the address of a nearby overlay router. The sender sends control information indicating the overlay group that is to be used and application-level information that describes the contents of the transmission. In addition, the sender publishes arbitrary flow description information, disseminated via MINT, which can be used by traffic management policies within the network infrastructure. To initiate communication, the sender and overlay router allocate and exchange transport connection

parameters (e.g., UDP or TCP port numbers) that the overlay router then uses to receive that group's traffic. Once this connection is established, the sender can inject traffic into the overlay network using unicast.

2.2.3 Transit VIFs

Once a packet has been successfully delivered to an overlay router either across an LVIF via multicast or via direct unicast communication, it is propagated to all other LVIFs and unicast receivers in the overlay network that include members interested in receiving traffic sent to the overlay group in question. To accomplish this, the first-hop overlay router prepends an overlay packet header on the UDP payload and forwards the traffic to peer overlay routers according to multicast "routes" that span transit virtual link interfaces (TVIF). Overlay routers forward the packet based on the overlay group stored in the overlay header. In effect, a TVIF provides a virtual interconnection between virtually adjacent overlay routers.

Two or more overlay routers peer with each other across a TVIF using two types of communication channels:

- a bi-directional TCP connection to exchange control messages (i.e., routing messages and group membership state), and

- a unidirectional, connectionless UDP channel to forward data packets.

We refer to these two abstractions as the control channel and data channel respectively.

The data channel may be either unicast (called a TVIF) or multicast (called a multicast TVIF), and in either case, packets are framed over UDP. In the unicast case, packets are simply transmitted to the adjacent peer using that peer's IP address and a well-known UDP port. That peer, in turn, receives the packet on the expected UDP port and inspects the overlay header to determine where to route the packet next, and so forth.

The Multicast TVIF

The multicast case is more complex and interesting. Here, a collection of overlay routers peer with each other using native multicast running across a single TVIF. In this case, the communication abstraction is isomorphic to a fully-connected mesh of overlay routers but with the efficiency of multicast. The control channels are effected using a fully-connected mesh of TCP connections, while the pairwise virtual data channels are effected using a single native multicast group. To isolate this multicast traffic to a well-confined region, peer routers may be placed in an overlay scope, where either or both administrative and TTL scope boundaries limit the reach of data traffic. This scope defines a specific segment of the overlay network and peer routers forward overlay packets to each other by embedding them in native multicast datagrams. To effect this, overlay routers map overlay addresses onto native group address using a well-defined hash function and the peers that are interested in receiving a certain overlay group join the corresponding native group at the network layer. In this fashion, overlay routers exploit native multicast routers across the transit regions in an efficient manner.

For example, suppose there are three routers A, B, and C, and overlay groups G1 and G2. Further suppose that the spanning tree for group G1 is incident to A and B and the spanning tree for G2 is incident to all three routers. Then, A and B would join group G1 where A, B, and C would all join group G2. Thus, when A sends packets to G1, they are sent only to B, and when anyone sends packets to G2, they are sent to everyone in this set.

However, overlay routers cannot natively join an overlay group. Instead, they hash the overlay group to a native group, where the hash function is chosen to map the entire overlay address range into the native multicast address range that is bound to the overlay scope of the multicast TVIF. Call the hash function that performs this mapping "h(.)". Thus, when an overlay router learns that it is incident to the multicast routing tree for some overlay group G with respect to a particular multicast TVIF, it joins the native multicast group h(G). In turn, any peer routers that forward packets for group G on that TVIF, will send the packet to group h(G) and the router in question will consequently receive the desired traffic. For example, A and B would exchange packets by joining and sending packets to group h(G1), whereas A, B, and C would each join group h(G2) and forward packets to each other over that native group. Note that each TVIF may have an h(.) that depends on the overlay scope used to define that TVIF.

Moreover, the overlay scope can be configured with an arbitrary address range so that multiple multicast TVIFs can overlap in non-trivial and interesting ways. Thus, a single router can be virtually attached to multiple, isolated multicast scopes and forward traffic among them.

For example, in the above scenario, A, B, and C form one TVIF, while C, D, and E might form another. In this case, if we ensure that the multicast address ranges for the overlay scopes that define two TVIFs are disjoint, then C can efficiently forward traffic between the two regions. In effect, two hash functions, say h1 and h2, would map overlay groups to native groups in the two TVIFs. Thus, C would forward a packet addressed for overlay group G from the first TVIF to the second TVIF by receiving packets sent to the native group h1(G) and forwarding those packets to the native group h2(G). (As explained later, the overlay router performs several important checks to ensure that the traffic is received in congruence with the routing state so as to avoid traffic loops and so forth.)

Because multiple overlay groups may, in general, hash to the same native multicast group, there is a potential for address collision. However, these sorts of collisions merely result in bandwidth inefficiency and do not cause packets to be incorrectly routed. The reason for this is that the forwarding logic in an overlay router is based on the overlay address carried in the overlay header, not on the native, network-layer address. Thus, the next-hop native multicast address need not be unique. The only adverse affect of a collision is that traffic may flow to an overlay router that has no interest in receiving that traffic thus wasting network bandwidth. In effect, we have a tension between the size of the address block used (and hence the amount of group-state stored in the native multicast routers) and the bandwidth-efficiency of the scheme.

2.2.4 Unicast-bank VIFs

To support clients without direct multicast connectivity (which may be the predominant form of interaction with an overlay network into the indefinite future), the overlay multicast service interface includes a mechanism whereby clients can communicate directly with an overlay router over UDP unicast. Here, an end-host contacts the overlay router through some application-specific fashion (e.g., a Web request for a video stream) and establishes group membership for a particular overlay group. The setup state, including the overlay group in question, is specified in the connection setup process, e.g., the overlay group could be embedded in a Web URL and HTTP could be used within an overlay router to accept requests from senders and receivers to attach to the overlay network.

Because of the transparency of the OMN infrastructure, end clients would not normally know how and when to connect to an overlay router. Thus, redirection can be used (as in HTTP) to redirect a given client away from the origin server that advertises streaming content, say, to a nearby overlay router. A content-aware redirection server can be used to map an IP address, for instance, to a nearby overlay router. Alternatively, in a reverse-proxy streaming configuration, the end client could be redirected to one of a large number of overlay routers at a centralized broadcast site arranged in a cluster for load balancing.

There are two types of unicast-bank VIFs (LVIF): unidirectional and bidirectional. In the unidirectional variant, clients cannot send traffic into the OMN (though they can send application-specific feedback to the incident overlay router), whereas in the bi-directional variant, clients can in fact do so. The former case is well matched to one-to-many applications like broadcast streaming media, while the latter case is appropriate for many-to-many applications like video conferencing.

As a practical concern, the unidirectional case scales more gracefully than the bi-directional case because the packet replication process can occur after the management and routing decision. That is, a unidirectional UVIF can simply "machine gun" a copy of a given packet to all the end-clients subscribed to the given group, rather than schedule and process each such packet individually (thus necessitating copying and allocating memory for this function). The bi-directional case, on the other hand, provides richer control because it allows for fully meshed intercommunication in a bandwidth-managed fashion.

Group membership is implied by the presence of the unicast end client. When the client requests attachment to a given overlay group, the incident overlay router treats the client as in the LVIF case and propagates group membership state as needed to initiate reception of traffic sent to the overlay group in question. The group membership state is timed out using various heuristics and application-level hints. For example, if the overlay router begins to receive port unreachable or host unreachable ICMP messages, then the flow will be terminated and the overlay group unsubscribed to (unless there are other hosts receiving said group's traffic through the UVIF). Alternatively, if the client uses TCP control connection to initiate the UDP flow—as does the Real-time Streaming Protocol (see, RFC2326), then the presence of the persistent TCP connection can be used to wire down the group state. Once the connection is closed or aborts, the state can be expired and the group relinquished.

3. Application Shims

Unlike the IP Multicast service model—where senders do nothing more than transmit packets to a group address—the ONIN forwarding infrastructure requires that a source announce its presence on a well-known, domain-wide control group (i.e., via DGMP) or signaled directly to an overlay router via SSP. If the end-hosts were OMN-aware, these control messages could be generated directly by the host. However, to maintain compatibility with existing applications and reduce the burden on application developers, the OMN architecture assumes that some other agent in the network can generate these messages on behalf of the sending application. An agent that provides this type of middleware glue is referred to as a "shim".

Applications like the Real Networks G2 Server, Microsoft Netshow, Cisco IP/TV, and the Mbone tools can bridge into an overlay network by building shims around the applications. The shims configure the tools appropriately and signal

overlay group setup information to a nearby overlay router using the techniques described above. A shim can be a non-trivial application and export a sophisticated user interface. For example, a broadcast control center could be built that provides the OMN signaling information to bridge between a bank of RealNetworks G2 servers and a cluster of overlay routers arranged in a mesh.

Note that there is no pressing need for a specialized shim on the receive side of an application. For example, if a receiver joins through a UVIF, the overlay router is directly informed of all required control information. Likewise, if an unmodified receiver application joins a multicast group, an agent on the attached LAN can snoop the IGMP traffic and relay appropriate DGMP signaling information to the designated overlay router in a generic fashion. That is, the receiver need only specify its presence and interest for a group and need not provide any additional signaling state. Shims are not necessary in all applications but can be used to perform a web redirection process, as desired.

4. The Plugin Framework

Because overlay routers are situated at points of administrative disconnect and bandwidth heterogeneity and because they are application-aware, they provide a strategic vantage point for carrying out traffic management that accounts for locally and globally defined administrative policies as well as the semantics of the underlying data flow. Traffic management in multicast is especially difficult because the receivers for a given group may be situated along a heterogeneous set of network paths thus preventing a source from simply sending a packet flow to all receivers at the same rate. To solve this problem, the OMN architecture includes a traffic shaping stage that is applied to each overlay group's packet stream before it is transmitted over a virtual link in the overlay network. To effect this functionality, a hierarchical class-based bandwidth allocation scheme apportions the available bandwidth across a set of application-level processing agents, called plugins, that manage each group's packets

The plugin framework transforms an overlay router into a flexible and extensible platform for migrating application-level functionality into the network in a safe and restricted fashion. Each media flow is bound to an application-level handler, called a plugin, that can flexibly transform, thin, or otherwise control the flow it manages. The plugin data path is tightly integrated with forwarding semantics of the application data and adheres to the policy constraints imposed by external management and configuration tools. In a nutshell, plugins are application-level entities that are situated in the forwarding path of the overlay routers. Feedback from the packet scheduler informs the plugin of congestion and/or bandwidth availability, thereby allowing the plugin to best utilize available network resources in a tightly controlled fashion.

For example, a plugin might perform stream thinning within the network according to the bandwidth throughput it attains from the bandwidth allocator. If a media flow is represented as a number of simulcasted sub-components, each at a different bitrate and corresponding quality, then the plugin could forward the maximum number of streams that the bandwidth policy permits, thereby accommodating bandwidth heterogeneity from within the network infrastructure.

To maximize the efficacy of the application-level adaptation capability afforded by the plugin framework, the scheduler that controls the transmission of packets across VIFs

explicitly communicates with the plugin to adjust its rate. By providing specific feedback as to what rate the plugin can expect to achieve on each output link, the plugin can adjust the rate of the flow it manages to fit into its allotment.

The plugin framework is extensible. As new applications are introduced into the network, the overlay broadcast infrastructure can be easily and incrementally upgraded by dynamically downloading plugins that handle the new traffic types as needed. Abstractly, plugins execute in a "sandboxed" process context and intercommunicate with the overlay router through a narrow application-programming interface called the Plugin API. We can think of the overlay router plugin abstraction as an analog to Web browser plugins. Just as a browser plugin is selected based on a Web object's Multi-Purpose Internet Mail Extensions ("MIME"—see RFCs 1521 and 1522) type, the overlay router plugin is selected according to the traffic flow type signaled via MINT.

Given the goals of this plugin bandwidth management framework, a number of problems must be solved. First, packets must be classified, that is assigned to a certain traffic category, so that traffic can be managed in a policy-oriented fashion. Second, bandwidth must be allocated and shared among application specific processing agents. Third, packets must be actively scheduled to ensure that the traffic class policies and bandwidth allocations are adhered to. Finally, bandwidth policies must be defined, administered, and attached to the virtual links in the network.

5. Routine

A collection of overlay routers forms a logical, overlay network that provides the conduit for efficiently distributing media flows using the multicast forwarding techniques described herein. However, in order to properly forward packets across the overlay network, the overlay routers must know how to route packets across the network such that all interested receivers receive a single copy of each packet and no traffic loops arise. To this end, the OMN architecture must carry out two fundamental routing tasks:

- the overlay network must compute efficient spanning-tree routes for multicasting packets from every potential source to every potential receiver, and
- the network must track group membership along the distribution tree to prevent multicast traffic from flowing where it otherwise is not needed.

Rather than invent new routing protocols from scratch, the OMN architecture leverages existing multicast routing technologies that compute spanning trees and track group membership in a scalable fashion, but at the application layer rather than the network layer. The core approach is based on the Border Gateway Multicast Protocol (BGMP), but we simplify the so-called "rendezvous problem" (see below) by explicitly including the rendezvous point in the upper 32 bits of the overlay group address. To scale the routing infrastructure, overlay routers may be clustered into routing "broadcast hubs", where the routers that comprise the hub are interconnected with a high-speed multicast-capable LAN. Hubs, in turn, are interconnected across the wide area. In this way, traffic can be load-balanced between wide area hubs by spreading groups (i.e., broadcast channels) across multiple overlay routers. Likewise, unicast join requests can be distributed evenly across a broadcast hub to balance the load of an arbitrary number of client viewers.

5.1 Multicast Routing Background

The fundamental problem in multicast routing is to build up state in the network that interconnects each source with

every interested receiver via some sort of routing tree. This is typically called the "rendezvous problem", i.e., how multicast receivers and sources find each other in a scalable distributed fashion. Roughly speaking, there are two fundamental approaches to this problem: (1) broadcast traffic everywhere and prune it back from regions of the network where there are no interested receivers, and (2) broadcast traffic toward a core and have receivers join a broadcast channel by sending control messages to that core.

Unfortunately, "broadcast and prune" is quite unscalable since traffic goes everywhere in the network. On the other hand, anchoring a tree at a core (or "rendezvous point") in the network is more scalable but has the disadvantage that the participants must know where the core is (or the network must maintain a mapping from multicast group addresses to cores). A number of ad hoc approaches for this have been proposed in the research literature (for CBT and SM-PIM), but BGMP takes a novel approach where multicast addresses are bound to source domains. In this model, the network maintains a distributed mapping between multicast addresses and source domains. A key novelty in BGMP is to represent this mapping as a set of routes. That is, each BGMP domain is configured with one or more blocks of multicast addresses and that BGMP domain advertises these blocks (via scalable prefixes) across the BRs using a routing protocol like M-BGP, a Border Gateway Protocol for exchanging routing information between gateway hosts. The effect is that each border router then knows the shortest path toward the "owner domain" of a given block of multicast addresses. Thus, when a border router receives a join message for a given group, it can consult this auxiliary routing table to propagate the join toward the source domain and record or modify the "local view" of the multicast spanning tree in a forwarding table (the Forwarding Information Base or FIB). In short, the M-BGP routing table is used exclusively to set up the bi-directional tree and once the tree state is established in the FIB, the border router consults only the FIB to make forwarding decisions. In BGMP terminology, the source domain is called the "root", the routing table of multicast address blocks is called the "multicast RIB", and the forwarding table constructed from join/leave BGMP messages (which are routed via the RIB) is called the "tree state table".

Given that this scheme relies upon each root domain owning some set of multicast address blocks, the natural question arises as to how these addresses are allocated in a decentralized, robust fashion. To this end, BGMP proposes that some companion protocol issue address blocks dynamically across domains in a hierarchical fashion. The Multicast Address Set Claim (MASC) protocol has been proposed to serve this function for BGMP. The basic model in MASC is to arrange domains into a hierarchy and have children domains request blocks of multicast addresses from their parents, which in turn requests larger blocks from their parents, and so on up to some top-level domain that owns the entire multicast address space. The address blocks are claimed and released using timeouts in a fashion similar to how Dynamic Host Configuration Protocol (DHCP—RFC2131) allocates temporary IP addresses in a local network.

Unfortunately, the dynamic approach to address allocation taken by MASC introduces substantial complexity and potentially hard-to-predict dynamics. It is also not clear how top-level addresses will be arranged or what incentives will be in place to prevent sub-domains from requesting too many addresses. Moreover, the architecture relies upon address allocation preemption, where if a parent domain

reclaims a block of addresses, all the applications that are using that address must perform reallocation, presumably in some synchronized fashion. In addition, this framework is complicated by the fact that address blocks will become fragmented over time as applications allocate and release addresses from the various available blocks. In short, maintaining a decentralized, robust address allocation architecture, especially in the face of intermittent connectivity that is so commonplace in the Internet, appears to be fragile and difficult to deploy. Fortunately, these problems can be quite easily overcome if the address architecture can be changed (and since we are building a new overlay network infrastructure, this is easily done). By using a 64-bit address for the broadcast channel address, the IP address of the core can appear explicitly as the upper 32-bits of the overlay address.

The beauty of embedding prefix-based root domains in a larger address is that the resulting scheme is completely compatible with the BGMP multicast RIB because the root domain prefix is simply treated as additional bits of the multicast address. And the M-BGP style aggregation of multicast address blocks only requires a route computation that effectively uses root domain identifiers instead of multicast address blocks (because they are one in the same). In short, the RIB is a routing table that gives shortest path routes to the rendezvous points and has no dependence per se on overlay group addresses.

In light of this discussion, the OMN architectural components for overlay multicast routing are based on:

- a path-state routing (PSR) protocol to compute shortest-path routes toward the rendezvous point, and
- an overlay group membership (OGMP) protocol based on BGMP to track group membership across the OMN.

5.2 Path-State Routing Protocol (PSR)

In order to effectively build the multicast spanning trees, each overlay node must know the next hop along the shortest path from that node to the rendezvous point. That is, spanning trees are incrementally built up as group membership requests (graft and prune messages) are sent toward the rendezvous point according to the group's route. Thus, overlay routers run a distributed routing algorithm to compute these shortest paths.

As in BGMP, the OMN architecture exploits a path-state routing protocol to compute such routes. Any node in the overlay network may be configured as the rendezvous point for some set of overlay address prefixes (i.e., which may be the unicast address prefixes of the particular set of hosts within its domain). Once so configured, a node advertises "reachability" to that rendezvous point via itself. In turn, its directly attached neighbors propagate this reachability information to its neighbors and so forth. As in BGP, each advertisement includes the entire path of nodes to which the route corresponds, which allows the system to easily detect and avoid potential routing loops, and allows administrative policies to control how routes are filtered and propagated. As described above, the OMN architecture accounts for and optimizes the case that transit domains are implemented using native multicast. That is, overlay routers forward packets to a group of peer routers simply by transmitting a single copy using native multicast. This, however, has no bearing on the path-state routing computation since this method of forwarding is isomorphic with that of unicasting a copy to each peer. And, in the overlay topology, each peering relationship across a multicast transit VIF is managed as if it were a separate, point-to-point connection.

5.3 Group Membership

When a host joins (or leaves) an overlay group at the edge of the network, group membership state must be reflected into the network in a scalable fashion. The Overlay Group Management Protocol (OGMP) carries out this function by tracking group membership requests at an overlay node and propagating them as necessary to the appropriate peers based on the PSR routing data base.

As in BGMP, the OMN architecture effects group membership through a graft/prune protocol. When a host joins a group, a join message is sent toward the rendezvous domain (which is explicitly given in the join request) using PSR routes. As soon as the join message reaches a router that is already on the tree, the message stops. The message is sent hop-by-hop toward the rendezvous point across a TCP control connection that manages the peering relationship of each pair of overlay nodes. As in BGP and BGMP, the use of TCP connections provides reliability and flow control and thereby simplifies the routing protocol.

Likewise, when a source joins a group, a join message is sent toward (and all the way to) the rendezvous point. This implies that traffic flows to the rendezvous point even if it is not necessary to do so to reach all interested receivers. A future version of the group membership protocols may be more sophisticated and avoid this condition.

5.4 Loop Avoidance

A standard problem in both unicast and multicast routing protocols is traffic loops that are caused by inconsistent views of the routing database at different points in the network. This problem could potentially be exacerbated by the application-level nature of the OMN architecture and the interaction of the application-level routing infrastructure with the underlying network-level multicast layer. In this section, we discuss this interaction and argue that the OMN architecture is in fact robust against looping pathologies.

One challenge posed by the OMN architecture is that the underlying multicast regions used to glue together distinct multicast transit VIFS might not be completely partitioned from one another simply because such isolation might be hard to configure (and in fact is difficult to automatically ensure). Thus, we could end up in a situation where an overlay router R receives a packet say from multicast TVIF V1 and forwards it to TVIF V2 (out a separate physical interface). But if for some reason these two domains overlap physically, then the packet forwarded to V2 would reappear on domain V1 and R might forward it again creating a routing loop.

Fortunately, the application-level nature of the OMN leads to an easy solution for this problem whereby we leverage the peering relationship that already exists among overlay routers. As such, a router forwards a packet only if it arrives from one of its peers within the appropriate transit VIF. That is, a router accepts a packet only if it came from a peer router that it expected it to come from (which it can check since, unlike network-layer multicast, the peer's IP address appears explicitly in the packet). Thus, in the case above, R would see that the second copy of the packet from VIF V1 had originated from itself and therefore drop it. Note that this scheme generalizes correctly to indirect loops involving multiple VIFs because the set of peers incident to a multicast VIF are configured into each overlay node.

The other case of concern is when an overlay router accepts a packet from a multicast leaf VIF. Here, the router must be sure that the source of the packet is an end host and

not another overlay router (that, for instance, might be leaking packets into the leaf domain coincidentally). Since the OMN architecture requires that senders explicitly signal their presence using SSE, the overlay router knows whether any given packet is from a legitimate sender in that leaf VIF. Note that this is another example where routing at the application layer provides a simple solution to hard problems—since the routers appear explicitly in the forwarding function we can easily detect loops using simple consistency checks.

5.5 Black Hole Avoidance

One of the major difficulties of interoperability between multicast routing protocols is to ensure that disparate routing entities agree on the overall multicast routing tree for any given group. That is, when a packet crosses a routing domain, it must enter the new domain at a point that is topologically matched to that domain's view of the distribution tree. Yet, if the outer-domain protocol has an alternate viewpoint, the packet arrives at the wrong location and is dropped. A substantial fraction of the complexity of the BGMP protocol specification has to do with making sure this mismatch is properly handled for the various multicast routing protocols that exist.

However, just as application-level routing made the loop avoidance problems easy, it likewise provides an easy solution for avoiding black holes. Because network layer addresses are rewritten on each hop through the OMN, there is no need for overlay routers to ensure that the leaf domain multicast routes are congruent with the OMN world view. That is, when a packet is injected into a leaf or transit multicast VIF, the IP source address corresponds to the overlay router not the original source. Thus, black holes cannot occur because this traffic is routed exclusively against local rules in the containing overlay scope, which are not susceptible to any wide-area state.

A disadvantage of this approach, however, is that the original source address does not survive in tact and thus a multicast receiver has no direct means to identify the address of the origin. Instead, the OMN model assumes that transport-level protocols either provide their own mechanisms for source identification (as in RTP) or that applications are agnostic to the origin address (as with streaming media players from Microsoft and RealNetworks).

Transaction Example in the OMN Architecture

Next, a description of hardware suitable for use with the present invention is presented, followed by a detailed walk-through of information transactions using the overlay approach of the present invention.

Description of Hardware

FIG. 3A is an illustration of computer system 1 including display 3 having display screen 5. Cabinet 7 houses standard computer components (not shown) such as a disk drive, CDROM drive, display adapter, network card, random access memory (RAM), central processing unit (CPU), and other components, subsystems and devices. User input devices such as mouse 11 having buttons 13, and keyboard 9 are shown. Other user input devices such as a trackball, touch-screen, digitizing tablet, etc. can be used. In general, the computer system is illustrative of but one type of computer system, such as a desktop computer, suitable for use with the present invention. Computers can be configured with many different hardware components and can be made in many dimensions and styles (e.g., laptop, palmtop,

pentop, server, workstation, mainframe). Any hardware platform suitable for performing the processing described herein is suitable for use with the present invention.

FIG. 3B illustrates subsystems that might typically be found in a computer such as computer 100.

In FIG. 3B, subsystems within box 20 are directly interfaced to internal bus 22. Such subsystems typically are contained within the computer system such as within cabinet 7 of FIG. 3. Subsystems include input/output (I/O) controller 24, System Random Access Memory (RAM) 26, Central Processing Unit (CPU) 28, Display Adapter 30, Serial Port 40, Fixed Disk 42 and Network Interface Adapter 44. The use of bus 22 allows each of the subsystems to transfer data among the subsystems and, most importantly, with the CPU. External devices can communicate with the CPU or other subsystems via bus 22 by interfacing with a subsystem on the bus. Monitor 46 connects to the bus through Display Adapter 30. A relative pointing device (RPD) such as a mouse connects through Serial Port 40. Some devices such as Keyboard 50 can communicate with the CPU by direct means without using the main data bus as, for example, via an interrupt controller and associated registers (not shown).

As with the external physical configuration shown in FIG. 3A, many subsystem configurations are possible. FIG. 3B is illustrative of but one suitable configuration. Subsystems, components or devices other than those shown in FIG. 3B can be added. A suitable computer system can be achieved without using all of the subsystems shown in FIG. 3B. For example, a standalone computer need not be coupled to a network so Network Interface 44 would not be required. Other subsystems such as a CDROM drive, graphics accelerator, etc. can be included in the configuration without affecting the performance of the system of the present invention.

FIG. 3C is a generalized diagram of a typical network.

In FIG. 3C, network system 160 includes several local networks coupled to the Internet. Although specific network protocols, physical layers, topologies, and other network properties are presented herein, the present invention is suitable for use with any network.

In FIG. 3C, computer USER1 is connected to Server1. This connection can be by a network such as Ethernet, Asynchronous Transfer Mode, IEEE standard 1553 bus, modem connection, Universal Serial Bus, etc. The communication link need not be a wire but can be infrared, radio wave transmission, etc. Server1 is coupled to the Internet. The Internet is shown symbolically as a collection of server routers 162. Note that the use of the Internet for distribution or communication of information is not strictly necessary to practice the present invention but is merely used to illustrate a preferred embodiment, below. Further, the use of server computers and the designation of server and client machines is not crucial to an implementation of the present invention. USER1 Computer can be connected directly to the Internet. Server1's connection to the Internet is typically by a relatively high bandwidth transmission medium such as a T1 or T3 line.

Similarly, other computers at 164 are shown utilizing a local network at a different location from USER1 computer. The computers at 164 are coupled to the Internet via Server2. USER3 and Server3 represent yet a third installation.

Note that the concepts of "client" and "server," as used in this application and the industry, are very loosely defined and, in fact, are not fixed with respect to machines or software processes executing on the machines. Typically, a

server is a machine or process that is providing information to another machine or process, i.e., the "client," that requests the information. In this respect, a computer or process can be acting as a client at one point in time (because it is requesting information) and can be acting as a server at another point in time (because it is providing information). Some computers are consistently referred to as "servers" because they usually act as a repository for a large amount of information that is often requested. For example, a World Wide Web (WWW, or simply, "Web") site is often hosted by a server computer with a large storage capacity, high-speed processor and Internet link having the ability to handle many high-bandwidth communication lines. A server machine will most likely not be manually operated by a human user on a continual basis, but, instead, has software for constantly, and automatically, responding to information requests. On the other hand, some machines, such as desktop computers, are typically thought of as client machines because they are primarily used to obtain information from the Internet for a user operating the machine.

Depending on the specific software executing at any point in time on these machines, the machine may actually be performing the role of a client or server, as the need may be. For example, a user's desktop computer can provide information to another desktop computer. Or a server may directly communicate with another server computer. Sometimes this is characterized as "peer-to-peer," communication. Although processes of the present invention, and the hardware executing the processes, may be characterized by language common to a discussion of the Internet (e.g., "client," "server," "peer") it should be apparent that software of the present invention can execute on any type of suitable hardware including networks other than the Internet. Although software of the present invention, such as the MediaBridge software, may be presented as a single entity, such software is readily able to be executed on multiple machines. That is, there may be multiple instances of a given software program, a single program may be executing on two or more processors in a distributed processing environment, parts of a single program may be executing on different physical machines, etc. Further, two different programs, such as a client and server program, can be executing in a single machine, or in different machines. A single program can be operating as a client for one information transaction and as a server for a different information transaction.

FIGS. 4A-C are next discussed to present two examples of initiating and maintaining an overlay multicast from a content source to a requesting destination computer. The two examples differ only in the manner in which the destination computer makes a request to be included as a recipient of the multicast content information. In the first example, the overlay multicast is initiated by a unicast request from the destination computer. In the second example the overlay multicast is initiated by a native multicast request from the destination computer.

In the Figures, an "M" in a box indicates a MediaBridge computer that handles the overlay routing of the invention as discussed above. An "R" in a box indicates a router, such as a typical router on an intranet, the Internet, or other network where the router manages the flow of information to, and from, a local, or otherwise defined relatively self-contained, network that is connected to other networks which are also managed by associated routers. In the Figures, one router is assumed to manage a single local area network (LAN) and one MediaBridge computer is assigned to each network that can receive overlay multicast information, or that can act as

a router for overlay multicast information, according to the present invention. MediaBridge computers are not strictly required on every LAN. Unicast attachment with the network allows a MediaBridge to be situated arbitrarily far from the client but, in general, the closer proximity of MediaBridges to clients aids in overlay routing. As discussed above, MediaBridge computers form an overlay network where the computers are in a peering, relationship with each other. MediaBridge computers maintain tables of overlay groups which are updated via a membership protocol. The tables are used to route information between the MediaBridge computers according to subscriptions, or memberships, to the overlay groups. Critical to the routing is a mapping of overlay groups to native groups by using the tables.

FIGS. 4A-C show more detail than in FIG. 2 by depicting individual end-user, or "host," computers, routers and MediaBridge computers; but otherwise show the same architecture as in FIG. 2. For example, group of networks 120 is a "transit domain" as depicted in FIG. 2 while the other LANs can be considered "leaf domains" as discussed above. Across transit domains native multicasting is preferably used to achieve overlay forwarding through unicast, peer-to-peer or other types of prior art routing can be used to achieve the overlay routing. Note that the specific topology, interconnections and number and type of devices shown in the Figures is only for purposes of illustrating the following examples, it should be readily apparent that many arrangements of computers, routers, wide or local area networks, physical links, or other processing devices or communication structures may be used with the invention.

FIG. 4A shows a unicast-initiated overlay routing aspect of the present invention.

In FIG. 4A, the goal is to have streaming media information, such as video program digital data including image and audio information, originate from source computer 100 to be received by destination computer 102. Naturally, many other destinations will exist in a typical multicast of information. The same mechanisms and techniques discussed here with respect to the single destination 102 can apply to any number of destinations. Because of the use of an overlay address and mapping protocol, the system of the present invention scales easily without requiring changes to existing network software and hardware.

Although the invention is discussed with respect to multicasting of streaming media information, any type of information can be distributed over a network by using the techniques of the present invention. For example, Internet distribution of software applications and updates, stock quotes, web pages, web cache updates, news, etc., can all be distributed more quickly, more efficiently, and with more control and monitoring by using the techniques of the present invention.

The present invention allows each potential recipient of a multicast to explicitly request the information. Such a request can be performed by the selection of a human user at the requesting computer, can be initiated automatically by a computer, or can be achieved by other means.

FIG. 4A shows the case where the receipt of multicast information is initiated by a unicast manner in response to a user's request. Specifically, destination 102 is a desktop computer operated by a user who is browsing web pages. In general, any type of computer running any number of applications, operating systems, user environment, etc., is suitable for use with the present invention. The web page that the user is currently viewing on the desktop computer is

"served" by web server computer 104. Web server 104 stores, and serves, information to other computers, such as destination computer 102, in the form of web page content, hyperlinks (i.e., uniform resource locators or "URLs") and other formats.

In the present example, the link of interest is a link to receive the streaming media video production which is to be provided by source computer 100. The data from source computer 100 may already be streaming when destination computer 102 makes a request for the stream. Preferably, source computer 100 registers its channel with the overlay network so that other MediaBridges and web servers "know" how to associate an overlay channel with the data stream. For example, a directory services table can be maintained by web server 104 so that web server 104 can carry out the redirection process for a published channel. When the user of destination computer 102 chooses to receive the video program, e.g., by clicking on a link, web page graphic, symbol or other control, web server 104 transfers information on how to subscribe to the video program as shown by the path 106. In the preferred embodiment, the transactions between destination computer 102 and web server 104 are governed by HTTP/TCP. However, it should be apparent that the invention is adaptable to a variety of different network communication protocols and standards.

Web server 104 returns the identification for MediaBridge computer 108 (referenced as M_1 in FIG. 4A) to destination computer 102. Web server 104 makes the decision to have destination computer 102 route through MediaBridge computer 108 since web server 104 is provided with information associating computers on the Internet with optimal MediaBridge computers. In the preferred embodiment, the optimal MediaBridge computer that a destination computer will use is generally the MediaBridge computer closest in proximity to the destination computer. Proximity can be a factor of both geographic distance and the electronic network path between the destination and MediaBridge computers.

In the present example, web server 104 is provided with information on making the association between a destination computer and a MediaBridge computer as, for example, where a table in web server 104 associates one or more destination computers with a specific MediaBridge computer. Note that it is possible to have such associations performed by another computer rather than the web server. For example, a MediaBridge, or other computer, connected on a local area network to web server 104 can be used. Also, the computer making the destination computer and MediaBridge computer association can be remote from web server 104, although the web server requires access to the mapping table to redirect destination computer 102 correctly.

In a first approach to connecting destination computer 102 to the media stream, web server computer 104 can provide an overlay channel identifier to destination computer 102. The channel identifier is used by the various MediaBridge computers to route the proper content to a destination computer. The channel identifier, or channel address, is 64 bits in the preferred embodiment. A channel name is used in the URL and is mapped to a corresponding channel ID as part of the redirection process. Optionally, other information can be provided by the web server. Such additional information can be used either by destination computer 102 to make the subscription request, or can be used by a MediaBridge computer to service subscription requests and to provide efficient multicast relaying. For example, statistics can be kept about the requesting user's

computer, geographic location, etc. This can be used for demographic analysis, to make predictions about the destination computer's ability to process data at a sufficient bandwidth, or for other reasons.

Once destination computer 102 has been provided with the appropriate MediaBridge computer address and channel identification information the destination computer makes a subscription request to MediaBridge computer 108.

FIG. 4B illustrates a next major step in the setup for overlay routing in the present example.

In FIG. 4B, once destination computer 102 acquires the subscription information from web host 104, destination computer uses the subscription information to send out one or more packets that indicate that MediaBridge computer 108 is to receive the subscribed channel. In the preferred embodiment, the subscription data includes an identification of the desired channel, i.e., the video program, that is to be received, the destination computer's identification (namely, destination computer 102), and other information. Note that the location of the MediaBridge computer can be different from that shown in FIG. 4B. Specifically, the MediaBridge computer can exist anywhere on the Internet and need not be part of the LAN that the destination computer is on.

Once MediaBridge 108 receives destination computer 102's subscription information, MediaBridge computer 108 uses the overlay channel ID to initiate the subscription process. The subscription process is a combination of the native multicasting network architecture and the overlay multicast architecture as described in detail, above. Ultimately, MediaBridge computer 108 sends one or more packets of information in an attempt to subscribe to the appropriate native multicast group. For purposes of the example, we assume that the appropriate native multicast group to which MediaBridge M2 will subscribe for purposes of handling the overlay routing with region 120 needed by the video program from source 100 to destination 102 is designated as "a." The overlay multicast group that is associated with the native multicast is designated as "A."

In the preferred embodiment, a direct 1-to-1 mapping of native to overlay groups is not possible since a native group uses 32 bits for an address and an overlay address is 64 bits. The upper-32 bits are used to specify a rendezvous point. A rendezvous point lookup algorithm can be locally configured, for example, into an agent that monitors IGMP messages. Another possibility is to have the rendezvous point selection algorithm provided in a fashion similar to the unicast case where the overlay group address is stored at a MediaBridge by a redirection process (discussed below) and where the client is instructed to natively join group "a."

A second approach to unicast-initiated overlay routing provides a channel name used in the original requesting URL to be mapped to a corresponding overlay group as part of the redirection process. When the client attempts to retrieve the resource identified by this URL, the server generates a response that redirects the client to MediaBridge 108. The client may or may not be explicitly aware of the protocols in use to effect overlay multicasting. If the client is overlay-multicast capable, the overlay group can be included in the response message and the client can connect to MediaBridge 108 using Overlay-multicast specific protocols. If, however, the client is not capable of participating in the overlay multicast protocols (e.g., because the client is an existing piece of software that has a large installed base), then the client can communicate with MediaBridge 108 using its existing, unmodified control protocol (e.g., Real-Time Streaming Protocol (RTSP) as specified in Internet-

draft for RTSP, Mar. 27, 1997). Since the overlay group may not be transportable through that existing protocol, yet is required by MediaBridge 108 to join the broadcast channel, the server (104) contacts MediaBridge 108 with the overlay channel to use before generating the response to the original Web request from the client (102). As part of the dialogue, the MediaBridge returns a "port" identifier to the server (104), upon which it then awaits the client-specific connection. This allows the server (104) to respond to the client with a message that redirects the client to the subsequent port, thereby ensuring that the requesting client is properly associated with the overlay group temporarily stored at MediaBridge 108.

FIG. 5 illustrates the second approach to unicast-initiated overlay routing.

In FIG. 5, which corresponds to FIGS. 4A-C, destination computer 102 makes request 160 to web server 104 via, for example, clicking on a hyperlink to obtain the media stream. Web server 104 deposits state at MediaBridge computer M1 representing the overlay group channel "A" and a streaming media URL, e.g., "rtsp://server.fastforward.com/live/olympics.rm", as shown by path 162. Next, M1 returns to web server 104 an identification that TCP port "p" is the port on which requesting computer 102 should connect as shown by path 164. Web server 104 responds to requesting computer 102 with a message that redirects requesting computer 102 to stream media from M1 over port "p" as shown by path 166. Requesting computer 102 initiates a streaming media protocol, such as RTSP, by contacting M1 over port "p" as shown by path 168. M1 sends a subscription request for group "A" to M2 as shown by path 170. Packets, or portions, of information sent from the content source will then be relayed via the overlay routing to M1 and to requesting computer 102.

The request to subscribe is transferred through router R₁ to other routers on the Internet. The subscription request installs a forwarding state in the MediaBridges that indicates which peer MediaBridges are part of the spanning tree for the given overlay channel. An example of a "domain," or "region," of routers and MediaBridge computers is shown at 120 in FIG. 4B. A region is simply a portion of the Internet, or other network, that is logically, geographically or, due to network traffic considerations, that provides efficiencies in multicast distribution when two or more MediaBridge computers implement an overlay network within the region. Within a given region, there is a mapping of each overlay channel to a single native multicast channel. In the preferred embodiment this occurs via a "hash" function, as described above. However, as information is propagated among regions there will typically be different native multicasting channels used for a given portion of information, or packet. For various reasons, the native and/or the overlay channel mappings for a particular multicast stream can change. For example overloading or failures can cause rerouting and remapping of channels.

Ultimately, the MediaBridge computer 108 sends one or more packets of information in an attempt to subscribe to the overlay group in question. The next few paragraphs describe an example of subscribing to an overlay group. The protocol for performing the subscription is called the overlay group membership protocol (OGMP). For the example, the overlay group is designated "A"; likewise, the symbol "a" denotes the native group that is computed from "A" using the overlay scope defined by region 120.

A subscription message is sent from MediaBridge 108 to the peer on the shortest path route, as determined by a

path-state routing computation, to the rendezvous point for "A". In this example, we assume the rendezvous point is MediaBridge M6 (it may be any MediaBridge in the overlay network, but it is most efficient to choose the rendezvous point near the broadcast source). Recall that the upper 32-bits of the overlay group represent the IP address of the rendezvous point.

In the configuration show in FIG. 4B, the peer on the shortest path from MediaBridge 108 to the rendezvous point (M6) is the MediaBridge labeled M2. Thus, MediaBridge 108 sends a "subscription request" for the overlay group in question to MediaBridge M2. In turn M2 propagates the subscription message to its peer that lies upon the shortest-path route to the rendezvous point, which in this case is MediaBridge M4. Likewise, M4 sends the request to M6. (These messages are sent reliably using TCP.) Each time a subscription message is received and processed by a MediaBridge, the MediaBridge records subscription information for the group in question and updates its forwarding information base so that when packets are received they can be properly routed to the peers subscribed to said overlay group.

In the case of a multicast transit VIF, the MediaBridge must additionally decide which native group to join using the prior art protocols as a function of the overlay group. For example, when M2 receives the subscription request, it joins the native multicast group "a" in addition to forwarding the subscription request for "A" onto M4. Thus, when M4 receive data packets addressed to overlay group "A", it will forward them into the overlay scope defined by region 120 by transmitting the overlay packet addressed to native group "a". Since M2 is has joined native group "a" using prior art protocols, it will receive said packet and in turn can forward that packet to its downstream peer M1.

After MediaBridge M₁ 108 has subscribed to the appropriate native multicast group, "a" in this example, it will receive native multicast transmissions for the group. Once MediaBridge computer 108 has joined the native multicast channel, it begins to receive information for the video program. This is illustrated in FIG. 4C by path 122. Although a specific routing is illustrated in FIG. 4C, naturally any number, and configuration, of routers, switches, servers or other devices can be used. Furthermore, the physical channel can be hardware, radio-frequency signals, infra red signals, fiber optic light signals, etc.

Once the overlay group, and group routing tables, information is distributed and stored, data transfer from source computer 100 can proceed accurately. Data is forwarded from source computer 100 to MediaBridge computer M₆ receives packets from source computer 100 and adds a header with the overlay group address to each packet. Each packet is then forwarded to M₄. M₄ multicasts the packets to the native group "a." M₂ receives the packets as M₂ is listening to native group "a" as a result of the association of "a" with "A" in the tables. M₂ forwards the packets to M₁. M₁ strips off the overlay header and forwards the native packet to the client, destination computer 102.

Naturally, any other computers on LAN 126 are also provided with the same packets if they have subscribed to overlay channel "A." Thus, this approach of overlaying a multicast channel designation onto an existing native multicast channel designation is able to benefit from native multicast according to the prior art. In addition, the act of using an overlay channel designation allows MediaBridge computers to regulate, and otherwise control, the transmission of packets associated with a specific stream to com-

puters on the same local network as the MediaBridge computer. Another advantage is that, where additional destination computers such as 124 are "downstream" from transit domain 120 and also desire to receive on overlay channel "A," duplication of packets to the transit domain, and over links within the transit domain, is not necessary. Thus, the broadcast scales efficiently because only one copy of each packet needs to be sent to the transit domain from the content source via MediaBridge M₆, and only one copy of each packet is transferred between MediaBridges within the transit domain.

Although the present example discusses a single MediaBridge computer associated with each local area network, other arrangements are possible. For example, a MediaBridge computer can be associated with more than one local area network where the networks are connected. Additionally, there may be more than one MediaBridge computer for a single local area network to provide fault tolerance and stability. The selection of the number of MediaBridge computers to use for a given number of networked computers, subnetworks, regions, domains, etc., is made based on the overall topology of the network area involved, traffic considerations, control requirements, etc. In general, there can be multiple transit domains interconnected in arbitrary ways by MediaBridges, or processors or processes performing one or more of the MediaBridge functions discussed herein.

As each MediaBridge computer receives information, or packets, designated for an overlay channel that the MediaBridge is participating in as a relay device, the MediaBridge computer checks an internally-stored table (i.e., the FIB) for the Internet protocol (IP) address of each peer machine that should receive the packet to ensure that the packet arrived from an acceptable peer. If so, the table indicates the IP addresses of additional peers to which the packet should be forwarded. It then transfers, or relays, copies of the packet to each of these machines. The preferred embodiment transfers the packets by using standard User Datagram Protocol (UDP). The efficiency of this distribution with respect to the present invention is largely determined by how a network administrator sets up the location of MediaBridges and the peer-to-peer tables in each MediaBridge computer. Naturally, one desirable scheme would minimize the amount of repetitious relays of the same packets to different MediaBridges by configuring the tables of peering relationships in a geographic, logical or network traffic sense, or in another arrangement designed to improve throughput, efficiency, control, monitoring or some other desirable goal. Efficiency of the overlay multicast system also depends on the extent to which native multicasting can be used to effect transfers.

Note that, that any point where a MediaBridge computer is in a store-and-forward position with respect to the streaming content that the MediaBridge computer can make decisions as to whether, and where, to route the packets. In other words, every time a MediaBridge computer is used to relay a packet, there can be a control mechanism for restricting, managing or modifying the relayed information. Aspects of the present invention relating to management and control of the media stream are discussed in detail in the co-pending patent application(s) referenced at the beginning of this specification.

Assume that the video program is a television program with commercial slots. MediaBridge computer M₂ can be used to insert a locally customized commercial into an appropriate commercial slot. MediaBridge computer M₁ can be used to restrict the video program entirely (e.g., a

pay-per-view presentation) from some computers and allow it to others in LAN 126. Or, where MediaBridge computer 108 determines that the bandwidth requirements of streaming the complete video program are too high for LAN 126, MediaBridge computer 108 can restrict the bandwidth of the video program, e.g., by reducing the image size, resolution, frame rate, color depth, etc. Other benefits are possible. For example, any MediaBridge can store, or cache all, or a portion of, the video program information. Caching decisions can be controlled by administratively specified bandwidth policies and traffic classes as described in co-pending patent application(s) referenced at the beginning of this specification.

In the discussion above, FIGS. 4A-C have illustrated a unicast-initiated overlay routing scenario. Another type of scenario is initiated in a multicast manner and is referred to as multicast-initiated overlay routing.

Multicast-initiated overlay routing differs from unicast-initiated overlay routing only in the manner that the destination computer subscribes to the content stream. In contrast to the unicast-initiated approach, the multicast-initiated approach allows a destination computer, such as destination computer 102 in FIGS. 4A-C, to make a request via a native multicast to join a particular native multicast group. For example, in the discussion above with respect to FIGS. 4A-C, destination computer 102 can make a multicast request to join native multicast group "a." MediaBridge computer 108 is programmed to detect such requests by monitoring IGMP traffic on the LAN, and processes the request by performing the steps described above to subscribe to native multicast group "a."

Table I, below, provides an overview summary with respect to data structures and how they are created and used in the examples discussed above in FIGS. 4A-C. Note that, although specific mechanisms, formats and uses are mentioned in relation to the data, that other possibilities are within the scope of the present invention. For example, tables can be constructed, modified, transferred and maintained (i.e., "processed") by manual means, or automatically by one or more processors or computers. The data and data structures can be processed by any prior art methods, systems and protocols or can be processed by specific new techniques as presented in this specification. The steps of Table I can be performed in a different order than the order shown in Table I. Also, all of the steps of Table I may not be necessary to practice the invention and additional steps can be included without exceeding the bounds of the invention.

TABLE I

1. A link registry is stored in, or made available to, R7. The link registry associates MediaBridge M1 with computers on M1's LAN, including destination computer 102.
2. Mapping tables associating M2, M3, M4 and M5 via a native multicasting channel are determined for purposes of native multicasting information within transit domain 120.
3. A request is made by content source 100 to register with the overlay network. This is handled by using an external directory source. The association between the content source and the overlay channel group designation can be transferred among MediaBridges.
4. Destination computers, such as destination computer 102 make requests for content information. Examples of how this is done include
 - (a) using a hyperlink to receive a URL from web server 104 that redirects destination computer to MediaBridge M1 according to the link registry at R7, where the redirection includes an identification of the content source such as by using the content source's overlay address, "A";
 - (b) destination computer 102 makes a unicast or multicast

TABLE I-continued

request of a MediaBridge, such as M1, for content by using the content source's native address, overlay address or other identification.

5. MediaBridge M1 sends a subscribe request for the content source information by using the overlay address "A" or other identification.

6. MediaBridge M2 receives the subscribe request and adds an association of overlay group address "A" with M1 so that packets from content source 100 received by M2 are sent to M1 and eventually to destination computer 102. Note that subscription can occur prior to, during, or even after content source 100 begins streaming data.

7. As M6 received packets from content source 100, M6 adds a header corresponding to the overlay group address "A" to each packet. Each packet with header "A" received by any of the MediaBridges is sent along the associated paths to other MediaBridges. This means that MediaBridges within the transit domain receive "A" packets via native multicast over channel "a". To achieve native multicast, the overlay address is included in the overlay header and carried in the native multicast packet. MediaBridges can add back the overlay address in the header for subsequent delivery to other MediaBridges. For example, M2's transmission of "A" packets to M1 includes the overlay address in the header. This allows M1 to continue overlay routing within M1's LAN of multiple overlay channels.

8. At each receipt of packets within a MediaBridge, benefits of bandwidth management, control, monitoring and other features through additional processing are possible as discussed herein and in co-pending patent applications referenced above.

FIG. 6 illustrates details of header and address processing in the present invention.

In FIG. 6, content source 202 sends information in the form of packets such as packet 204. Packet 204 includes an IP header 206 having a destination field and source field. The destination field indicates that the packet is destined for MediaBridge M1 and that the source for the packet is S. The packet data is contained in a UDP format "payload" 208. When MediaBridge computer M1 received the packet, it changes the destination and source indications to M2 and M1, respectively. Additionally, an overlay header is inserted between the IP header and the payload. This packet is shown at 210. The overlay channel indication is "A" in the overlay header, which is also in UDP format.

Packet 210 is received by MediaBridge computer M2. M2 is part of a native multicast group and so is able to distribute the packet via native multicast over the native multicast channel "a." Accordingly, M2 changes the destination and source indicators in the native header to "a" and M2, respectively. Packet 212 is then transmitted throughout multicast domain 214 where it is received by M3 and M4. MediaBridges such as M5 which haven't joined native multicast group "a" do not receive packet 212. MediaBridge M4 uses the overlay channel designation "A" to send the packet to client R1 after stripping off the overlay header "A" so that the packet appears to R1 as a standard packet. M3 and M4 both check the source address and overlay group of packet 212 to ensure that it came from an appropriate peer (in this case M2). If not, the packet would have been dropped.

Additional routing of the packet is performed by M3 by the use of a second native multicasting domain 222 using native multicast address "b." M3 uses native multicast group "b" by specifying the destination of packet 220 (having the same payload as packet 212) as "b." Thus, multiple different native multicast groups can be used to distribute the same overlay channel. Packet 220 is distributed through domain 222 via native multicast channel "b" to be received by M6 and other possible MediaBridges, routers, servers, computers, etc. (not shown) that are subscribed to native

multicast channel "b." M6, similar to M4's operation, uses the overlay channel designation "A" to determine that the packet should be sent to R2 and R3. M6 first strips off the overlay channel information before sending the packet to R2 and R3.

Although the invention has been presented with respect to particular embodiments thereof, these embodiments merely illustrate possible embodiments of the invention, the scope of which is determined solely by the appended claims.

What is claimed is:

1. An overlay routing processor for transferring information over a computer network, wherein the computer network has a native routing protocol that defines computers as members of native groups for purposes of routing information among members of a given native group, the overlay routing processor comprising

- instructions for associating computers on the network with a given overlay group;
- instructions for determining whether received information is associated with the given overlay group;
- instructions for routing the received information to the computers associated with the given overlay group by using the native routing protocol;
- instructions for identifying a specific native group as an efficient distribution channel for the given overlay group,

wherein the instructions for routing include instructions for using the specific native group to perform the routing and wherein the instructions for identifying a specific native group include instructions for using a hash function to perform the identification.

2. An overlay routing processor for transferring information over a computer network, wherein the computer network has a native routing protocol that defines computers as members of native groups for purposes of routing information among members of a given native group, wherein multiple overlay processors are coupled together over the network, the overlay routing processor comprising

- instructions for associating computers on the network with a given overlay group,
- instructions for associating computers on the network with a given overlay group;
- instructions for determining whether received information is associated with the given overlay group;
- instructions for routing the received information to the computers associated with the given overlay group by using the native routing protocol;
- instructions for identifying a specific native group as an efficient distribution channel for the given overlay group,

wherein the instructions for routing include instructions for using the specific native group to perform the routing;

a data table accessed by the processor for defining peer relationships between overlay processors; and

wherein the instructions for routing include instructions for using the defined peer relationships between overlay processors to perform the routing.

3. An overlay routing processor for transferring information over a computer network, wherein the computer net-

work has a native routing protocol, wherein an end-user computer is coupled to the network, wherein a first media information source is coupled to the network for sending media information to the network, the overlay routing processor comprising

- instructions for associating computers on the network with a given overlay group;
- instructions for determining whether received information is associated with the given overlay group;
- instructions for routing the received information to the computers associated with the given overlay group by using the native routing protocol;
- a data structure associating the media information with a first overlay channel identifier;
- instructions for receiving a request from the end-user computer to receive the media information;
- instructions for retrieving the first overlay channel identifier from the data structure and for associating the first overlay channel identifier with the request; and
- instructions for routing all or a portion of the media information received by the overlay routing processor to the end-user computer.

4. The overlay routing processor of claim 3, wherein a second media information source is coupled to the network for sending media information to the network, the overlay routing processor further comprising

- instructions for indicating an error condition if a second overlay channel identifier associated with the second media information source is the same as the first overlay channel identifier.

5. An overlay routing processor for transferring information over a computer network, wherein the computer network has a native routing protocol that defines computers as members of native groups for purposes of routing information among members of a given native group, comprising

- instructions for associating computers on the network with a given overlay group;
- instructions for determining whether received information is associated with the given overlay group;
- instructions for routing the received information to the computers associated with the given overlay group by using the native routing protocol;
- instructions for identifying a specific native group as an efficient distribution channel for the given overlay group,

wherein the instructions for routing include instructions for using the specific native group to perform the routing;

- instructions for associating a native group with an overlay group; and

instructions for changing the association between an overlay group and a native group.

6. The overlay routing processor of claim 5, wherein the association between an overlay group and a native group includes defining a range of native multicast addresses.

7. The overlay routing processor of claim 6, wherein the range of native multicast addresses is defined as an IP₄ Class D address and a prefix length.

* * * * *

05/05/04

2155

Attorney Docket No. 030048001US

Express Mail No. EV336677851US

9/A
250
PATENT 5-20-04
entered



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: FRED B. HOLT *ET AL.*

APPLICATION No.: 09/629,576

FILED: JULY 31, 2000

FOR: **BROADCASTING NETWORK**

EXAMINER: YOUNG N. WON

ART UNIT: 2155

CONF. No: 5408

Amendment Under 37 C.F.R. § 1.111

RECEIVED

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

MAY 07 2004
Technology Center 2100

Sir:

The present communication responds to the Office Action dated February 4, 2004 in the above-identified application. Please amend the application as follows:

Amendments to the Specification begin on page 2.

Amendments to the Claims are reflected in the listing of claims beginning on page 6.

Remarks begin on page 13.

A

Amendments to the Specification:

In accordance with 37 CFR 1.72(b), an abstract of the disclosure has been included on page 3. In accordance with 37 CFR 1.73, a brief summary of the invention has been included on page 4. In addition, the status of the related cases listed on page 1 of the specification has been updated and can be found on page 5.

ABSTRACT

A technique for broadcasting data across a network is provided. An originating participant sends data to another participant, which in turn sends the data that it receives from a neighbor participant to its other neighbor participants. Communication in the broadcast network is controlled by a contact module that locates the neighbor participants to which the seeking participant can be connected and by a join module that establishes the connection between the neighbor participants and the seeking participant. Data is numbered sequentially so that data that is received out of order can be queued and rearranged.

SUMMARY OF THE INVENTION

Embodiments of the invention deal with a non-routing table based method for broadcasting messages in a network. More specifically, a network in which each participant has at least three neighbor participants broadcasts data through each of its connections to neighbor participants, which in turn send the data that it receives to its other neighbor participants. The data is numbered sequentially so that data that is received out of order can be queued and rearranged.

A-2
Communication within the broadcast channel is controlled by a contact module and by a join module. The contact module locates a portal computer and requests the located portal computer to provide an indication of neighbor participants to which the participant can be connected. The join module receives the indication of the neighbor participants and establishes a connection between the seeking participant and each of the indicated neighbor participants.

Each participant in the network is connected to neighbor participants, and the participants and connections between them form an m -regular graph, where m is greater than 2. In addition, when a participant receives data from a neighbor participant, it sends the data to its other neighbor participants.

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to ~~U.S. Patent Application No. 09/629,576, entitled "BROADCASTING NETWORK," filed on July 31, 2000 (Attorney Docket No. 030048004 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,570, entitled "JOINING A BROADCAST CHANNEL," filed on July 31, 2000 (Attorney Docket No. 030048002 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,577, "LEAVING A BROADCAST CHANNEL," filed on July 31, 2000 (Attorney Docket No. 030048003 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,575, entitled "BROADCASTING ON A BROADCAST CHANNEL," filed on July 31, 2000 (Attorney Docket No. 030048004 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,572, entitled "CONTACTING A BROADCAST CHANNEL," filed on July 31, 2000 (Attorney Docket No. 030048005 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,023, entitled "DISTRIBUTED AUCTION SYSTEM," filed on July 31, 2000 (Attorney Docket No. 030048006 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,043, entitled "AN INFORMATION DELIVERY SERVICE," filed on July 31, 2000 (Attorney Docket No. 030048007 US);~~ ^(mygr) U.S. Patent Application No. ~~09/629,024, entitled "DISTRIBUTED CONFERENCING SYSTEM," filed on July 31, 2000 (Attorney Docket No. 030048008 US);~~ ^(mygr) and U.S. Patent Application No. ~~09/629,042, entitled "DISTRIBUTED GAME ENVIRONMENT," filed on July 31, 2000 (Attorney Docket No. 030048009 US),~~ ^(mygr) the disclosures of which are incorporated herein by reference. ^{currently patented (mygr)}

13

currently patented (mygr)

(mygr)

now under appeal (mygr)

currently patented

(mygr)

currently patented (mygr)

45

A

Amendments to the Claims:

Following is a complete listing of the claims pending in the application, as amended:

sub B1

1. (Currently Amended) A non-routing table based computer network having a plurality of participants, each participant having connections to at least three neighbor participants, wherein an originating participant sends data to the other participants by sending the data through each of its connections to its neighbor participants, ~~and~~ wherein each participant sends data that it receives from a neighbor participant to its other neighbor participants, and wherein data is numbered sequentially so that data received out of order can be queued and rearranged.

HY

2. (Original) The computer network of claim 1 wherein each participant is connected to 4 other participants.

3. (Original) The computer network of claim 1 wherein each participant is connected to an even number of other participants.

4. (Original) The computer network of claim 1 wherein the network is m-regular, where m is the number of neighbor participants of each participant.

5. (Original) The computer network of claim 4 wherein the network is m-connected, where m is the number of neighbor participants of each participant.

6. (Original) The computer network of claim 1 wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant.

4
7.

(Original) The computer network of claim 1 wherein all the participants are peers.

5
8. (Original) The computer network of claim 1 wherein the connections are peer-to-peer connections.

6
9. (Original) The computer network of claim 1 wherein the connections are TCP/IP connections.

7
10. (Original) The computer network of claim 1 wherein each participant is a process executing on a computer.

8
11. (Original) The computer network of claim 1 wherein a computer hosts more than one participant.

9
12. (Original) The computer network of claim 1 wherein each participant sends to each of its neighbors only one copy of the data.

13. (Currently Amended) A non-routing table based component for controlling communications of a participant with a broadcast channel, comprising:
a contact module that locates a portal computer and requests the located portal computer to provide an indication of neighbor participants to which the participant can be connected, wherein a connection between the portal computer and the participant is not established, and wherein a connection between the portal computer and the neighbor participants is not established; and
a join module that receives the indication of neighbor participants and establishes a connection between the participant and each of the indicated neighbor participants.

14. (Original) The component of claim 13 wherein each participant is a computer process.

15. (Original) The component of claim 13 wherein the indicated participants are computer processes executing on different computer systems.

16. (Original) The component of claim 13 including:
a broadcast module that receives data from a neighbor participant of the participant and transmits the received data to the other neighbor participants.

17. (Original) The component of claim 13 including:
a connection request module that receives a request to connect to another participant, disconnects from a neighbor participant, and connects to the other participant.

18. (Original) The component of claim 13 wherein the connections are established using the TCP/IP protocol.

14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

19. (Currently Amended) A non-routing table based broadcast channel for participants, comprising:

a communications network that provides peer-to-peer communications between the participants connected to the broadcast channel; and

for each participant connected to the broadcast channel, an indication of four neighbor participants of that participant; and

a broadcast component that receives data from a neighbor participant using the communications network and that sends the received data to its other neighbor participants to effect the broadcasting of the data to each participant of the broadcast channel, wherein data is numbered sequentially so that data received out of order can be queued and rearranged.

11
20. (Original) The broadcast channel of claim 19 wherein the broadcast component disregards received data that it has already sent to its neighbor participants.

¹²~~21~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein a participant connects to the broadcast channel by contacting a participant already connected to the broadcast channel.

¹³~~22~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein each participant is a computer process.

¹⁴~~23~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein each participant is a computer thread.

¹⁵~~24~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein each participant is a computer.

¹⁶~~25~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein the communications network uses TCP/IP protocol.

¹⁷~~26~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein the communications network is the Internet.

¹⁸~~27~~. (Original) The broadcast channel of claim ~~19~~¹⁰ wherein the participants are peers.

28. (Currently Amended) A non-routing table based broadcast channel comprising a plurality of participants, each participant being connected to neighbor participants, the participants and connections between them forming an m-regular graph, where m is greater than 2 and the number of participants is greater than m.

29. (Original) The broadcast channel of claim 28 wherein the graph is m-connected.

30. (Original) The broadcast channel of claim 28 wherein m is even.

31. (Original) The broadcast channel of claim 28 wherein m is odd and the number of participants is even.

32. (Original) The broadcast channel of claim 28 wherein the participants are computer processes.

33. (Original) The broadcast channel of claim 28 wherein the participants are computers.

34. (Original) The broadcast channel of claim 28 wherein the connections are established using TCP/IP protocol

35. (Original) The broadcast channel of claim 28 wherein a message is broadcast on the broadcast channel by an originating participant sending the message to each of its neighbor participants and by each participant upon receiving a message from a neighbor participant sending the message to its other neighbor participants.

36. (Currently Amended) A non-routing table based broadcast channel comprising a plurality of participants, each participant being connected to neighbor participants, the participants and connections between them form an m -regular graph, where m is greater than 2, ~~and~~ wherein when a participant receives data from a neighbor participant, it sends the data to its other neighbor participants, and wherein data is numbered sequentially so that data received out of order can be queued and rearranged.

37. (Original) The broadcast channel of claim 36 wherein the number of participants is greater than m .

38. (Original) The broadcast channel of claim 36 wherein the graph is m -connected.

39. (Original) The broadcast channel of claim 36 wherein m is even.

40. (Original) The broadcast channel of claim 36 wherein m is odd and the number of participants is even.

41. (Original) The broadcast channel of claim 36 wherein the participants are computer processes.

42. (Original) The broadcast channel of claim 36 wherein the participants are computers.

43. (Original) The broadcast channel of claim 36 wherein the connections are established using TCP/IP/protocol.

44. (Currently Amended) A non-routing table based computer-readable medium containing instructions for controlling communications of a participant of a broadcast channel, by a method comprising:

locating a portal computer;

requesting the located portal computer to provide an indication of neighbor participants to which the participant can be connected;

receiving the indications of the neighbor participants; and

establishing a connection between the participant and each of the indicated neighbor participants, wherein a connection between the portal computer and the participant is not established, and wherein a connection between the portal computer and the neighbor participants is not established.

A4

Sub 831

²⁰
~~45.~~ (Original) The computer-readable medium of claim ¹⁹~~44~~ wherein each participant is a computer process.

²¹
~~46.~~ (Original) The computer-readable medium of claim ¹⁹~~44~~ wherein the indicated participants are computer processes executing on different computer systems.

²²
~~47.~~ (Original) The computer-readable medium of claim ¹⁹~~44~~ including:

receiving data from a neighbor participant of the participant; and transmitting the received data to the other neighbor participants.

²³
~~48.~~

(Original) The computer-readable medium of claim ¹⁹~~44~~ including:

receiving a request to connect to another participant;
disconnecting from a neighbor participant; and
connecting to the other participant.

²⁴
~~49.~~

(Original) The computer-readable medium of claim ¹⁹~~44~~ wherein the connections are established using the TCP/IP protocol.

A4

REMARKS

Reconsideration and withdrawal of the rejections set forth in the Office Action dated February 4, 2004 are respectfully requested.

I. Rejections under 35 U.S.C. § 102

A. The Applied Art

U.S. Patent No. 6,611,872 to McCanne (*McCanne*) is directed to an overlay protocol and system for allowing multicast routing in the Internet to be performed at the application level. The overlay protocol uses routing tables to route information. Column 2, lines 45-49 and Column 23, lines 11-15. The overlay protocol fails to disclose the use of a portal computer to add new participants to a network. In addition, the overlay protocol fails to disclose a method in which data is numbered sequentially so that messages received out of order can be queued and rearranged.

B. Analysis

Distinctions between independent claims 1, 13, 19, 28, 36, and 44 and *McCanne* will first be discussed, followed by distinctions between *McCanne* and the remaining dependent claims.

As noted above, *McCanne* discloses an overlay protocol that uses routing tables to route information. Column 2, lines 45-49 and Column 23, lines 11-15. *McCanne* fails to disclose a non-routing table based method for routing information. Independent claims 1, 13, 19, 28, 36, and 44 have been amended to clarify the inherent language of previously pending claims 1, 13, 19, 28, 36, and 44. In other words, claims 1, 13, 19, 28, 36, and 44 has been amended to recite, among other limitations, a "non-routing table based" method for routing information. *McCanne* fails to disclose such a method for routing information. For at least this reason, claims 1, 13, 19, 28, 36, and 44 are patentable over *McCanne*.

McCanne fails to disclose a method by which "data is numbered sequentially so that data received out of order can be queued and rearranged". Independent claims 1, 19, and 36 have been amended to clarify the inherent language of previously pending claims 1, 19, and 36. In other words, claims 1, 19, and 36 have been amended to recite, among other limitations, a method by which "data is numbered sequentially so

that data received out of order can be queued and rearranged". *McCanne* fails to disclose such a method for numbering data. For at least this reason, claims 1, 19, and 36 are patentable over *McCanne*.

McCanne fails to disclose the use of a portal computer to locate neighbor participants for the seeking participant to connect to. In addition, *McCanne* fails to disclose a method in which "a connection between the portal computer and the participant is not established, and wherein a connection between the portal computer and the neighbor participants is not established". *McCanne* discloses a method in which an overlay router, not a portal computer, determines what receivers are present. Column 8, lines 53-56. In addition, *McCanne* discloses a method in which the overlay router joins the corresponding group. The embodiments of the invention disclose a method by which the portal computer does not join the neighbor participants. Independent claims 13 and 44 have been amended to clarify the inherent language of previously pending claims 13 and 44. In other words, claims 13 and 44 have been amended to recite, among other limitations, a method in which "a connection between the portal computer and the participant is not established, and wherein a connection between the portal computer and the neighbor participants is not established". *McCanne* fails to disclose such a method. For at least this reason, claims 13 and 44 are patentable over *McCanne*.

As is known, to anticipate a claim under 35 U.S.C. § 102, the reference must teach every element of the claim.¹ *McCanne* fails to disclose every limitation recited in independent claims 1, 13, 19, 28, 36, and 44. Since independent claims 1, 13, 19, 28, 36, and 44 are allowable, based on at least the above reasons, the claims that depend on independent claims 1, 13, 19, 28, 36, and 44 are likewise allowable. Thus, for at

¹ MPEP section 2131, p. 70 (Feb. 2003, Rev. 1). See also, *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1462 (Bd. Pat. App. & Interf. 1990) (to establish a *prima facie* case of anticipation, the Examiner must identify where "each and every facet of the claimed invention is disclosed in the applied reference."); *Glaverbel Société Anonyme v. Northlake Mktg. & Supply, Inc.*, 45 F.3d 1550, 1554 (Fed. Cir. 1995) (anticipation requires that each claim element must be identical to a corresponding element in the applied reference); *Atlas Powder Co. v. E.I. duPont De Nemours*, 750 F.2d 1569, 1574 (1984) (the failure to mention "a claimed element (in) a prior art reference is enough to negate anticipation by that reference").

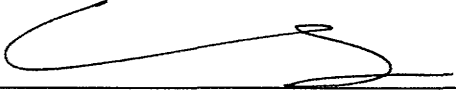
least this reason, claims 2-12, 14-18, 20-27, 29-35, 37-43, and 45-49 are patentable over *McCanne*.

II. Conclusion

In view of the foregoing, the claims pending in the application comply with the requirements of 35 U.S.C. § 112 and patentably define over the applied art. A Notice of Allowance is, therefore, respectfully requested. If the Examiner has any questions or believes a telephone conference would expedite prosecution of this application, the Examiner is encouraged to call the undersigned at (206) 359-8000.

Respectfully submitted,
Perkins Coie LLP

Date: 5/4/04


Chun M. Ng
Registration No. 36,878

Correspondence Address:

Customer No. 25096
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000



PTO/SB/21 (02-04)

Approved for use through 07/31/2006. OMB 0651-0031

U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

<h1>TRANSMITTAL FORM</h1> <p><i>(to be used for all correspondence after initial filing)</i></p>	Application Number	09/629,576-Conf. #5408	
	Filing Date	July 31, 2000	
	First Named Inventor	Fred B. Holt	
	Art Unit	2155	
	Examiner Name	Won, Young N.	
Total Number of Pages in This Submission	18	Attorney Docket Number	030048001US

RECEIVED
MAY 07 2004
Technology Center 2100

ENCLOSURES (Check all that apply)		
<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input checked="" type="checkbox"/> Amendment/Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Drawing(s) <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance communication to Technology Center (TC) <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to TC (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Other Enclosure(s) (please identify below): Return Postcard
		Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	PERKINS COIE LLP Chun M. Ng - 36,878
Signature	
Date	5/4/04

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV336677851US, in an envelope addressed to: Commissioner for Patents, P.O. Box 1456, Alexandria, VA 22313-1450, on the date shown below.	
Dated: 5/4/2004	Signature: (Melody Almberg)

OTAP JC61
MAY 04 2004
PATENT & TRADEMARK

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL

for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT (\$)**0**

Complete if Known	
Express Mail No.	EV336677851US
Application Number	09/629,576
Filing Date	July 31, 2000
First Named Inventor	Fred B. Holt
Examiner Name	Young N. Won
Art Unit	2155
Attorney Docket No.	030048001US

RECEIVED

MAY 07 2004

Technology Center 2100

METHOD OF PAYMENT (check all that apply)

Check Credit card Money Order Other None

Deposit Account:
 Deposit Account Number: 50-0665
 Deposit Account Name: Perkins Coie LLP

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments
 Charge any additional fee(s) during the pendency of this application
 Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description	Fee Paid
Fee Code	Fee (\$)	Fee Code	Fee (\$)		
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English Specification	
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$)**0**

FEE CALCULATION

1. BASIC FILING FEE

Large Entity		Small Entity		Fee Description	Fee Paid
Fee Code	Fee (\$)	Fee Code	Fee (\$)		
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1205	160	2005	80	Provisional filing fee	

SUBTOTAL (1) (\$)**0**

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims		Extra Claims		Fee from below		Fee Paid	
Independent Claims	49	- 49** =		X		=	0
Multiple Dependent	6	- 6** =		X		=	0

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) (\$)**0**

***or number previously paid, if greater; For Reissues, see above*

SUBMITTED BY		(Complete if applicable)	
Name (Print/Type)	Chun Ng	Registration No. (Attorney/Agent)	36-878
Signature		Telephone	206-359-8000
		Date	5/4/04

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Complete and send this form, together with applicable fee(s), to: **Mail**

Express Mail No. EV528705967US

SEP 15 2004

Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All other correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

25096 7590 07/13/2004

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

09/16/2004 CNGUYEN1 00000030 500665 09629576

01 FC:1501 1330.00 OP
02 FC:8001 6.00 OP
03 FC:1501 300.00 PA

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

Melody J. Almberg	(Depositor's name)
<i>Melody J. Almberg</i>	(Signature)
9/15/2004	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/629,576	07/31/2000	Fred B. Holt	030048001	5408

TITLE OF INVENTION: BROADCASTING NETWORK

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	\$1330	\$0	\$1330	10/13/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
WON, MICHAEL YOUNG	2155	709-204000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
 Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
 "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list
 (1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
 (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

Perkins Coie LLP

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)
 PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: The Boeing Company
 (B) RESIDENCE: (CITY and STATE OR COUNTRY) Seattle, Washington

Please check the appropriate assignee category or categories (will not be printed on the patent); individual corporation or other private group entity government

4a. The following fee(s) are enclosed:
 Issue Fee
 Publication Fee (No small entity discount permitted)
 Advance Order - # of Copies 2

4b. Payment of Fee(s):
 A check in the amount of the fee(s) is enclosed.
 Payment by credit card. Form PTO-2038 is attached.
 The Director is hereby authorized by charge ~~the enclosed~~ ^{any additional} fee(s), or credit any overpayment, to Deposit Account Number 50-0665 (enclose an extra copy of this form).

5. Change in Entity Status (from status indicated above)
 a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27. b. Applicant is not claiming SMALL ENTITY status. See, e.g., 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
 NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

(Authorized Signature) _____ (Date) 9/15/04

This collection of information is required by 37 CFR 1.341. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMIT THIS FORM WITH FEE(S)

P20

Notice of Allowability

Application No.

09/629,576

Examiner

Michael Y Won

Applicant(s)

HOLT ET AL.

Art Unit

2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

- 1. This communication is responsive to Amendment A and telephonic interview (attached herewith).
- 2. The allowed claim(s) is/are 1-3,7-12,19-27 and 44-49.
- 3. The drawings filed on _____ are accepted by the Examiner.
- 4. Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 - a) All b) Some* c) None of the:
 - 1. Certified copies of the priority documents have been received.
 - 2. Certified copies of the priority documents have been received in Application No. _____.
 - 3. Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.**

- 5. A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
 - 6. CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
 - (a) including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
 - 1) hereto or 2) to Paper No./Mail Date _____.
 - (b) including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.
- Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
- 7. DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

- 1. Notice of References Cited (PTO-892)
- 2. Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3. Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
- 4. Examiner's Comment Regarding Requirement for Deposit of Biological Material
- 5. Notice of Informal Patent Application (PTO-152)
- 6. Interview Summary (PTO-413), Paper No./Mail Date attached.
- 7. Examiner's Amendment/Comment
- 8. Examiner's Statement of Reasons for Allowance
- 9. Other _____.

Art Unit: 2155

EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Chun M. Ng (Reg. No. 36,878) on July 2, 2004.

2. The application has been amended as follows:

A. Claims 4-6, 13-18, and 28-43 have been cancelled.

B. Claims 1, 19 and 44 have been amended.

B' 1. (Currently Amended) A non-routing table based computer network having a plurality of participants, each participant having connections to at least three neighbor participants, wherein an originating participant sends data to the other participants by sending the data through each of its connections to its neighbor participants, wherein each participant sends data that it receives from a neighbor participant to its other neighbor participants, [and] wherein data is numbered sequentially so that data received out of order can be queued and rearranged, further

B

Art Unit: 2155

B¹

wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.

¹⁰
~~19.~~

(Currently Amended)

A non-routing table based broadcast channel

for participants, comprising:

a communications network that provides peer-to-peer communications between the participants connected to the broadcast channel; and

B²

for each participant connected to the broadcast channel, an indication of four neighbor participants of that participant; and

a broadcast component that receives data from a neighbor participant using the communications network and that sends the received data to its other neighbor participants to effect the broadcasting of the data to each participant of the to broadcast channel, [wherein data is numbered sequentially so that data received out of order can be queued and rearranged] wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.

BA

B

Art Unit: 2155

19
44.

(Currently Amended)

A non-routing table based computer-readable

medium containing instructions for controlling communications of a participant of a broadcast channel within a network, by a method comprising:

locating a portal computer;

requesting the located portal computer to provide an indication of neighbor participants to which the participant can be connected;

receiving the indications of the neighbor participants; and

establishing a connection between the participant and each of the indicated neighbor participants, wherein a connection between the portal computer and the participant is not established, wherein a connection between the portal computer and the neighbor participants is not established, further wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.

3. The following is an examiner's statement of reasons for allowance:

Claims 1-3, 7-12, 19-27, and 44-49 are pending.

Prior art of record McCanne (US 6,611,872 B1) does not disclose, teach, or suggest the claim limitation of wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein

Art Unit: 2155

the number of participants is at least two greater than m thus resulting in a non-complete graph as recited in the amended claims 1, 19, and 44, therefore the claims are found allowable over prior art of record.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Drawings

4. The drawings filed on July 31, 2000 are acceptable subject to correction of the informalities indicated on the attached "Notice of Draftsperson's Patent Drawing Review," PTO-948. In order to avoid abandonment of this application, correction is required in reply to the Office action. The correction will not be held in abeyance.

5. Fig.25 is missing. The numbering of figures should be consecutive. In order to avoid abandonment of this application, correction is required in reply to the Office action. The correction will not be held in abeyance.

6. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(4) because reference character "01-17" has been used to designate all figures from Fig.6- Fig.34 (example: reference character "01" for Fig.6 should be referenced by "601", likewise, reference character "01" for Fig.34 should be referenced "3401"). Corrected

6

Art Unit: 2155

drawing sheets are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet(s) should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

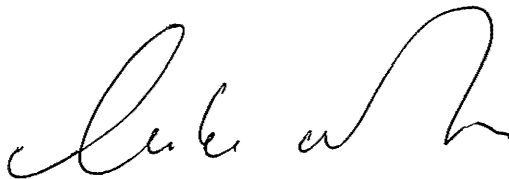
7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael Y Won whose telephone number is 703-605-4241. The examiner can normally be reached on M-Th: 6AM-3PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hosain T Alam can be reached on 703-308-6662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Art Unit: 2155

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Michael Y Won

A handwritten signature in black ink, appearing to read 'Michael Y Won', written in a cursive style.

July 12, 2004



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

25096 7590 07/13/2004
PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

EXAMINER

WON, MICHAEL YOUNG

ART UNIT PAPER NUMBER

2155

DATE MAILED: 07/13/2004

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
09/629,576 07/31/2000 Fred B. Holt 030048001 5408

TITLE OF INVENTION: BROADCASTING NETWORK

Table with 6 columns: APPLN. TYPE, SMALL ENTITY, ISSUE FEE, PUBLICATION FEE, TOTAL FEE(S) DUE, DATE DUE
nonprovisional NO \$1330 \$0 \$1330 10/13/2004

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

- A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.
B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

- A. Pay TOTAL FEE(S) DUE shown above, or
B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: **Mail** **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

25096 7590 07/13/2004

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/629,576	07/31/2000	Fred B. Holt	030048001	5408

TITLE OF INVENTION: BROADCASTING NETWORK

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	\$1330	\$0	\$1330	10/13/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
WON, MICHAEL YOUNG	2155	709-204000

<p>1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).</p> <p><input type="checkbox"/> Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.</p> <p><input type="checkbox"/> "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.</p>	<p>2. For printing on the patent front page, list</p> <p>(1) the names of up to 3 registered patent attorneys or agents OR, alternatively, _____ 1</p> <p>(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed. _____ 2</p> <p>_____ 3</p>
--	---

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE _____ (B) RESIDENCE: (CITY and STATE OR COUNTRY) _____

Please check the appropriate assignee category or categories (will not be printed on the patent); individual corporation or other private group entity government

<p>4a. The following fee(s) are enclosed:</p> <p><input type="checkbox"/> Issue Fee</p> <p><input type="checkbox"/> Publication Fee (No small entity discount permitted)</p> <p><input type="checkbox"/> Advance Order - # of Copies _____</p>	<p>4b. Payment of Fee(s):</p> <p><input type="checkbox"/> A check in the amount of the fee(s) is enclosed.</p> <p><input type="checkbox"/> Payment by credit card. Form PTO-2038 is attached.</p> <p><input type="checkbox"/> The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).</p>
--	---

5. Change in Entity Status (from status indicated above)

a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27. b. Applicant is not claiming SMALL ENTITY status. See, e.g., 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

(Authorized Signature) _____ (Date) _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMIT THIS FORM WITH FEE(S)



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/629,576	07/31/2000	Fred B. Holt	030048001	5408

25096 7590 07/13/2004

PERKINS COIE LLP
PATENT-SEA
P.O. BOX 1247
SEATTLE, WA 98111-1247

EXAMINER

WON, MICHAEL YOUNG

ART UNIT	PAPER NUMBER
2155	10

2155

10

DATE MAILED: 07/13/2004

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 857 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 857 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (<http://pair.uspto.gov>).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (703) 305-1383. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.



#11
10-30-04
K

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Air Mail No. EV528705967US, in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.
Dated: 9/15/04 Signature: *Melody Amberg*
(Melody Amberg)

Docket No.: 030048001US
Client Ref No. 99-481

(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Holt et al.
Application No.: 09/629,576
Filed: July 31, 2000
For: BROADCASTING NETWORK

Allowed: July 13, 2004
Confirmation No.: 5408
Art Unit: 2155
Examiner: Y. N. Won

**COMMENTS ON STATEMENT OF REASONS
FOR ALLOWANCE UNDER 37 CFR §1.104(E)**

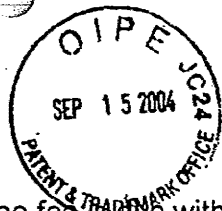
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicant has received the Examiner's Statement of Reasons for Allowance with the July 13, 2004 Notices of Allowance and Allowability regarding the above-identified application. Entry of the Statement into the record should not be construed as any agreement with or acquiescence in the reasoning stated by the Examiner. Each of the claims stands on its own merits and is patentable because of the combination it recites and not because of the presence or absence of any one particular element.

The Examiner's Statement was not prepared by Applicant and only contains the Examiner's possible positions in one or more reasons for allowability. Thus, any interpretation with respect to the Examiner's Statement of Reasons for Allowance should not be imputed to the Applicant.

Application No.: 09/629,576



Docket No.: 030048001US

Applicant believes no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 50-0665, under Order No. 030048001US from which the undersigned is authorized to draw.

Dated: _____

9/15/04

Respectfully submitted,

By _____

Chun M. Ng

Registration No.: 36,878

PERKINS COIE LLP

P.O. Box 1247

Seattle, Washington 98111-1247

(206) 359-8000

(206) 359-7198 (Fax)

Attorneys for Applicant

8-11-04

BPH
93
#12
W

Attorney Docket No. 030048001US
Client Ref No. 99-481

Express Mail Label EV336669033US



PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: FRED B. HOLT *ET AL.*

EXAMINER: M. Y. WON

APPLICATION NO.: 09/629,576

ART UNIT: 2155

FILED: JULY 31, 2000

CONF. NO: 5408

FOR: BROADCASTING NETWORK

Transmittal of Formal Drawings

MS Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

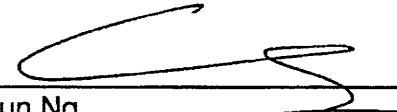
Sir:

Further to the Notice of Allowance dated July 13, 2004, and the Notice of Draftsperson's Patent Drawing Review dated July 8, 2004, enclosed are the required formal drawings, Figs 1-24 and 26-34 (39 Sheets).

No fees are believed due in connection with this response. However, if fees are due, the Commissioner is requested to charge them to Deposit Account No. 50-0665.

Respectfully submitted,
Perkins Coie LLP

Date: 8/10/04


Chun Ng
Registration No. 36,878

Correspondence Address:

Customer No. 25096
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000

09/629,576

6829634

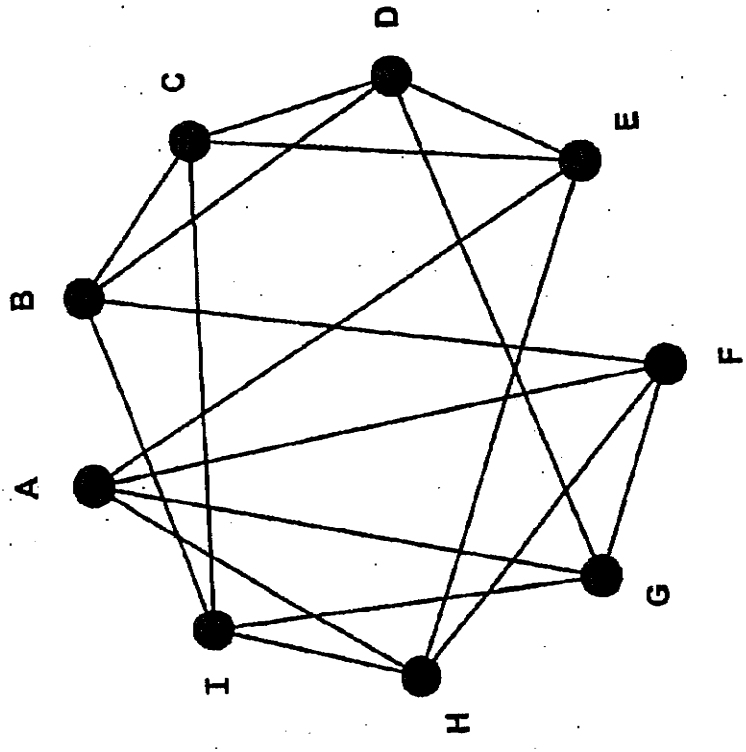


Fig. 1

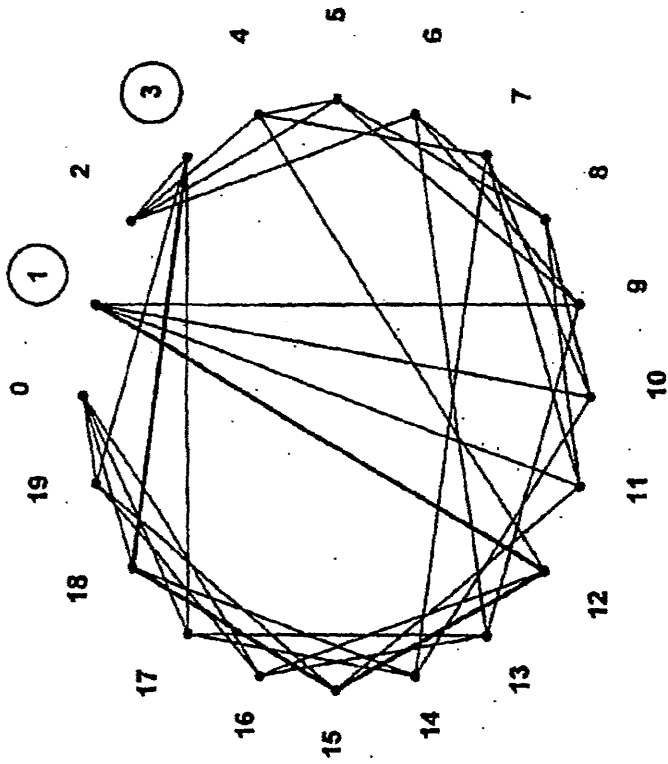


Fig. 2

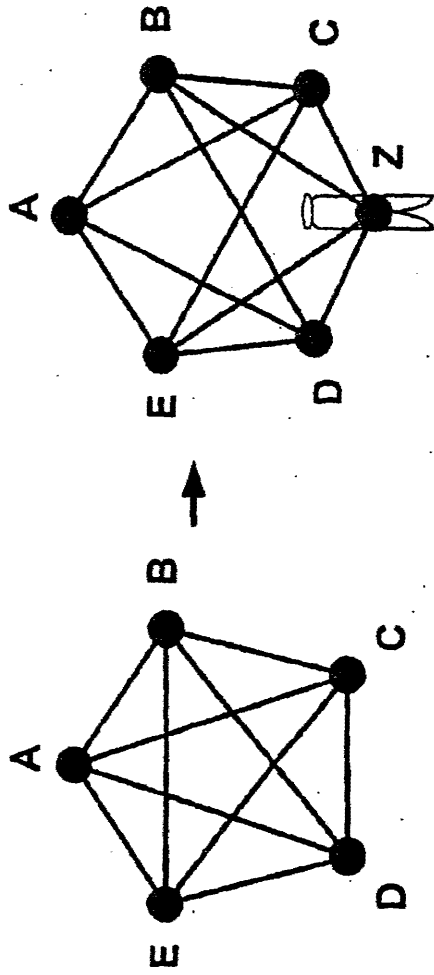


Fig. 3B

Fig. 3A

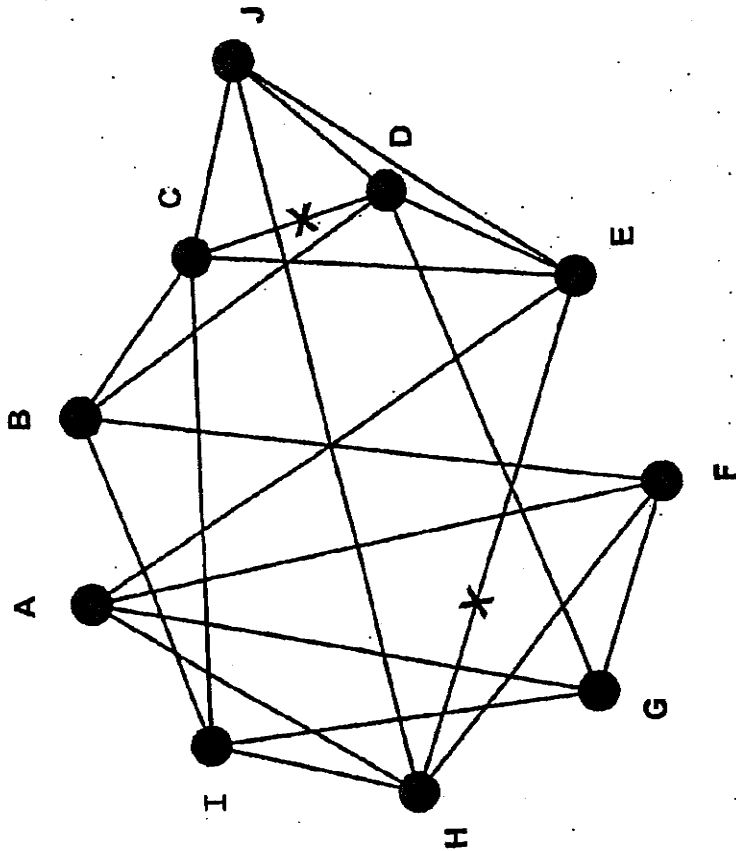


Fig. 4A

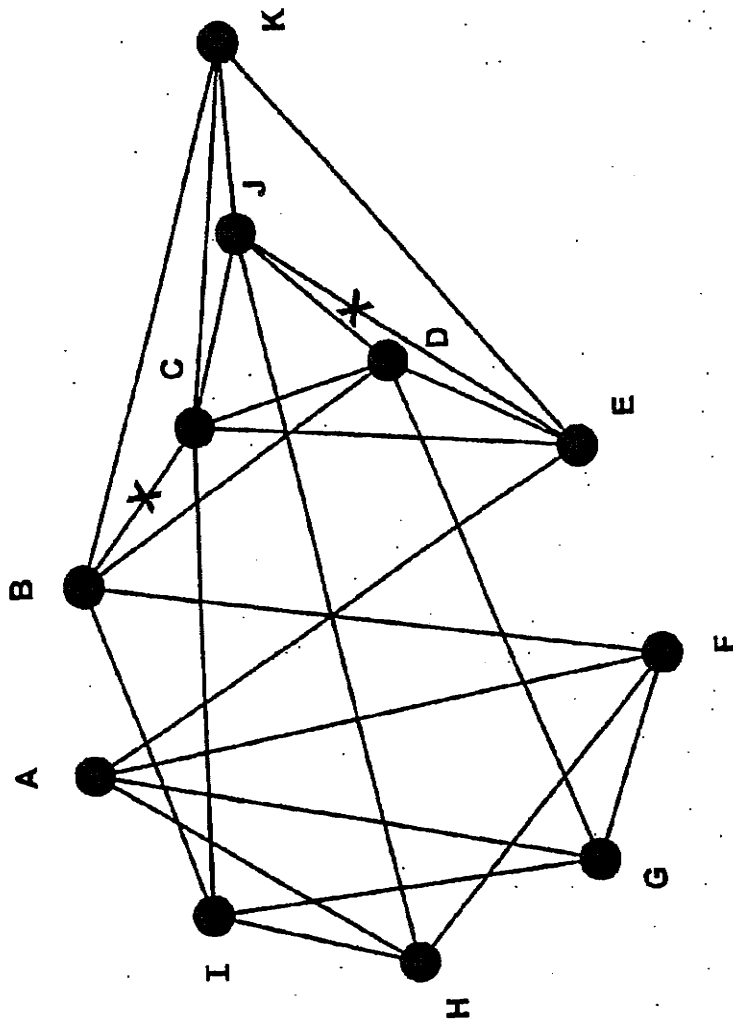


Fig. 4B

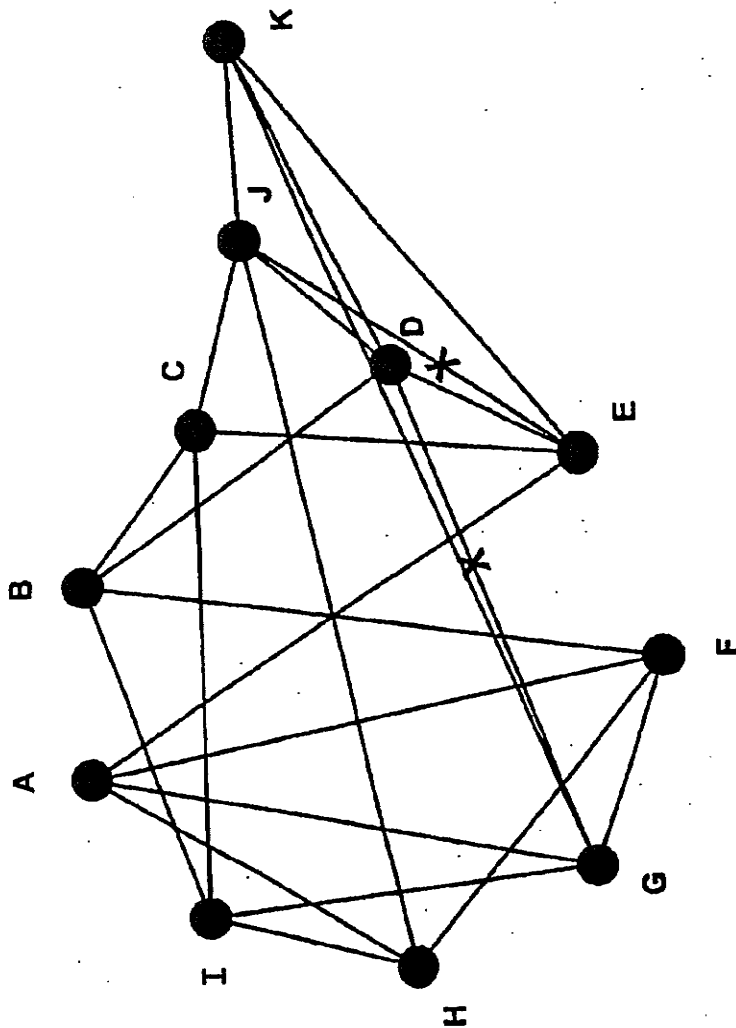


Fig. 4C

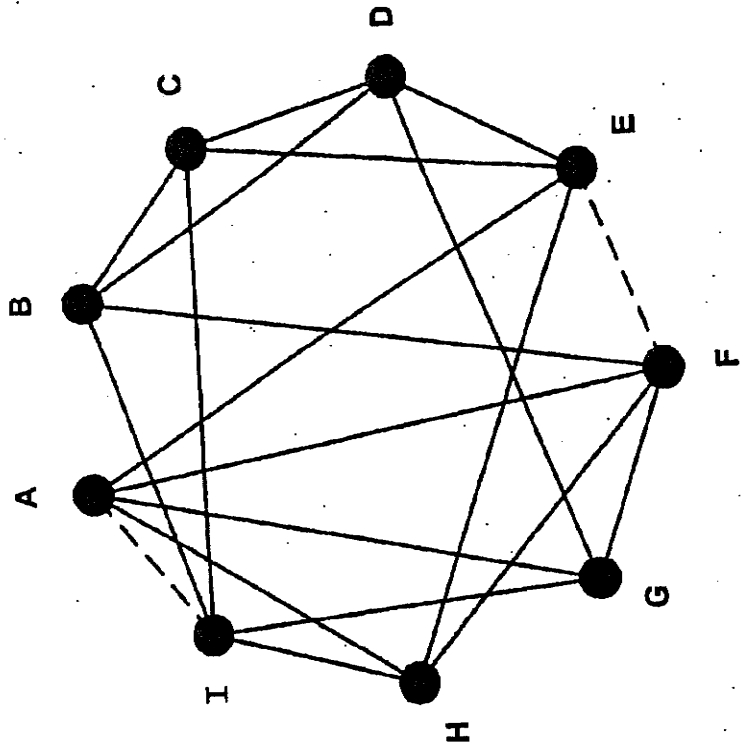


Fig. 5A

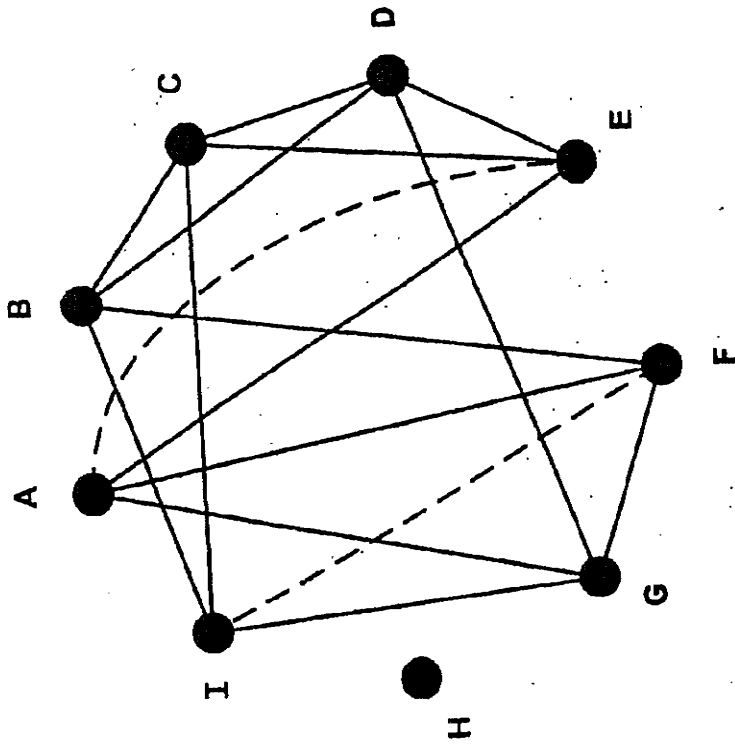


Fig. 5B

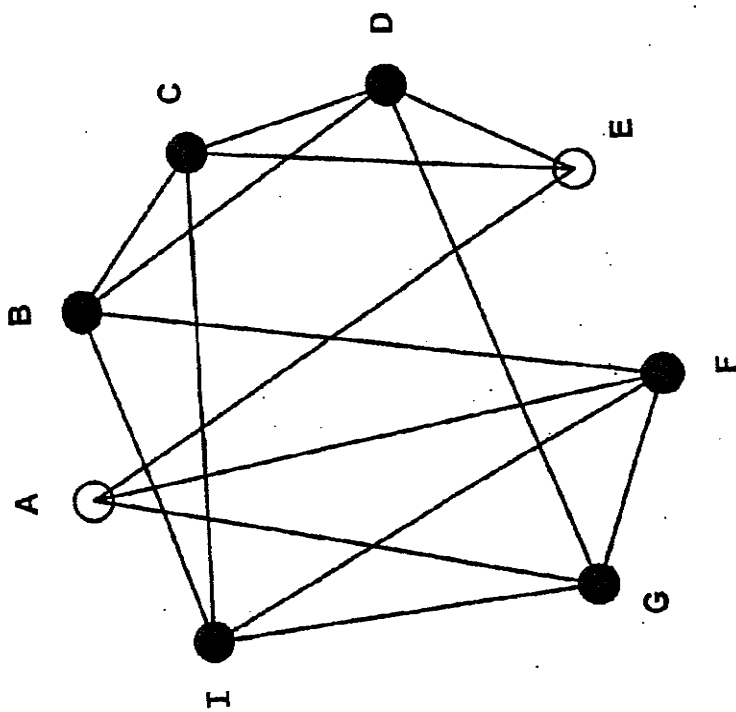


Fig. 5C

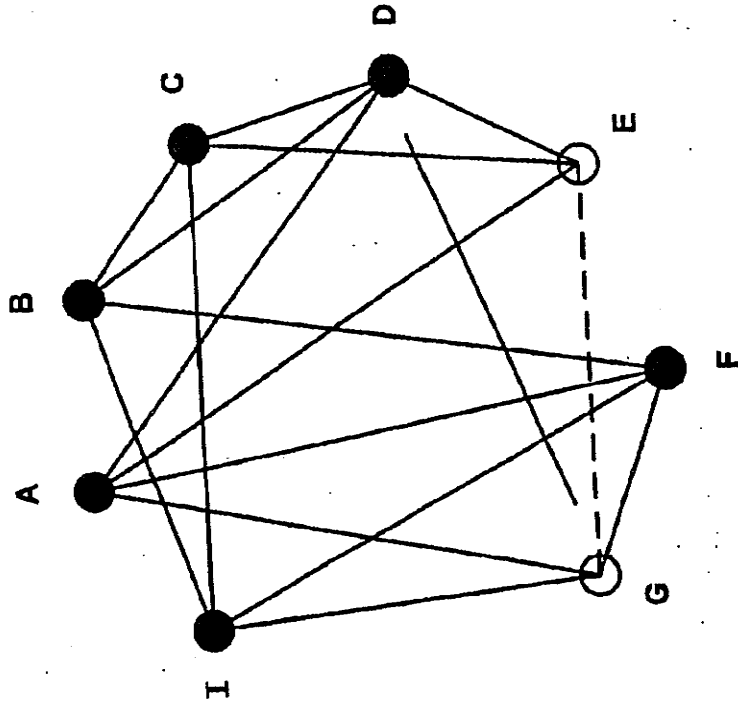


Fig. 5D

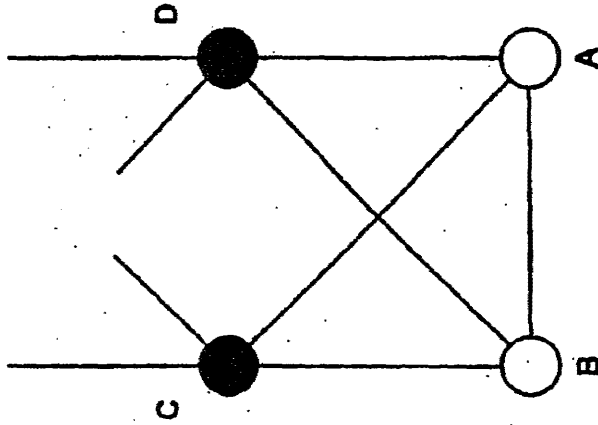


Fig. 5F

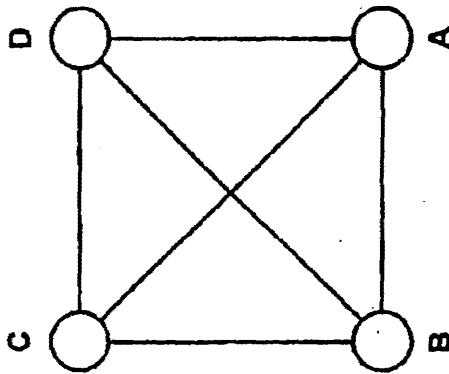


Fig. 5E

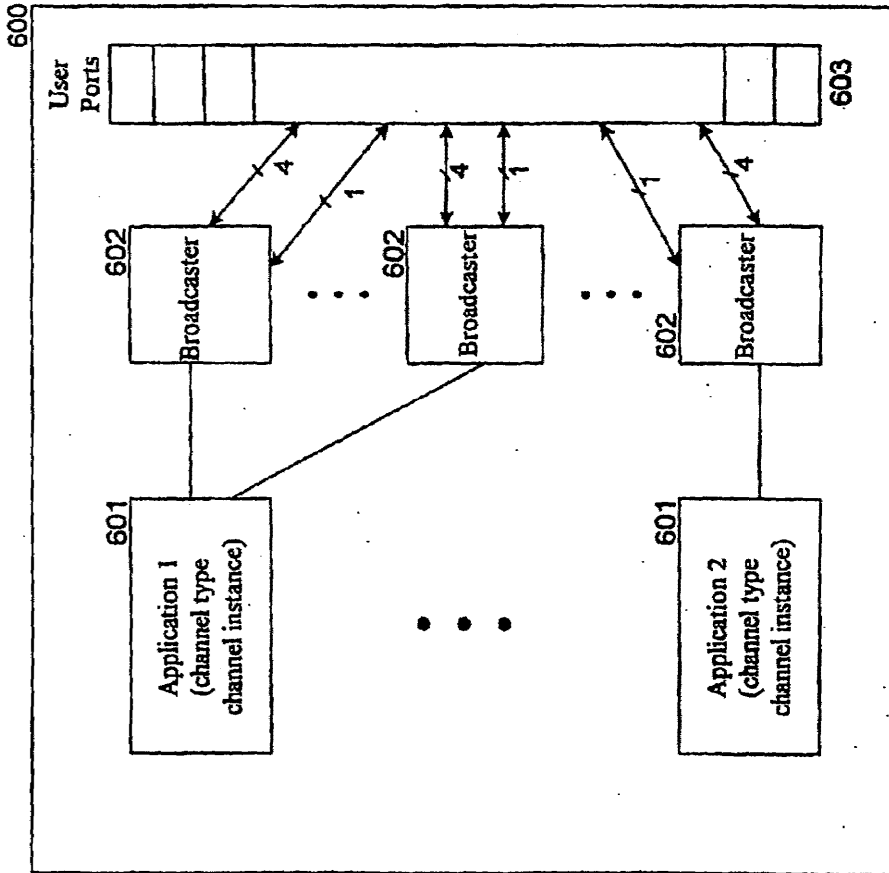


Fig. 6

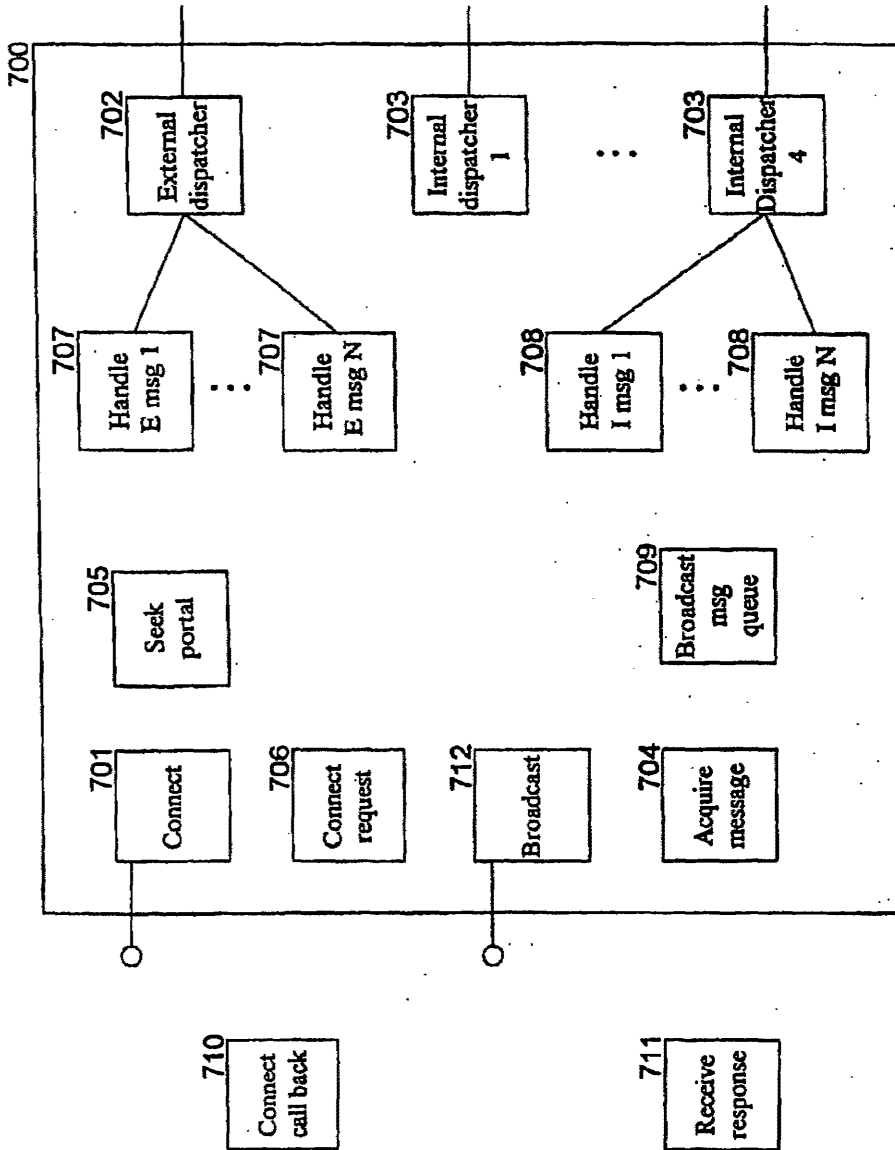
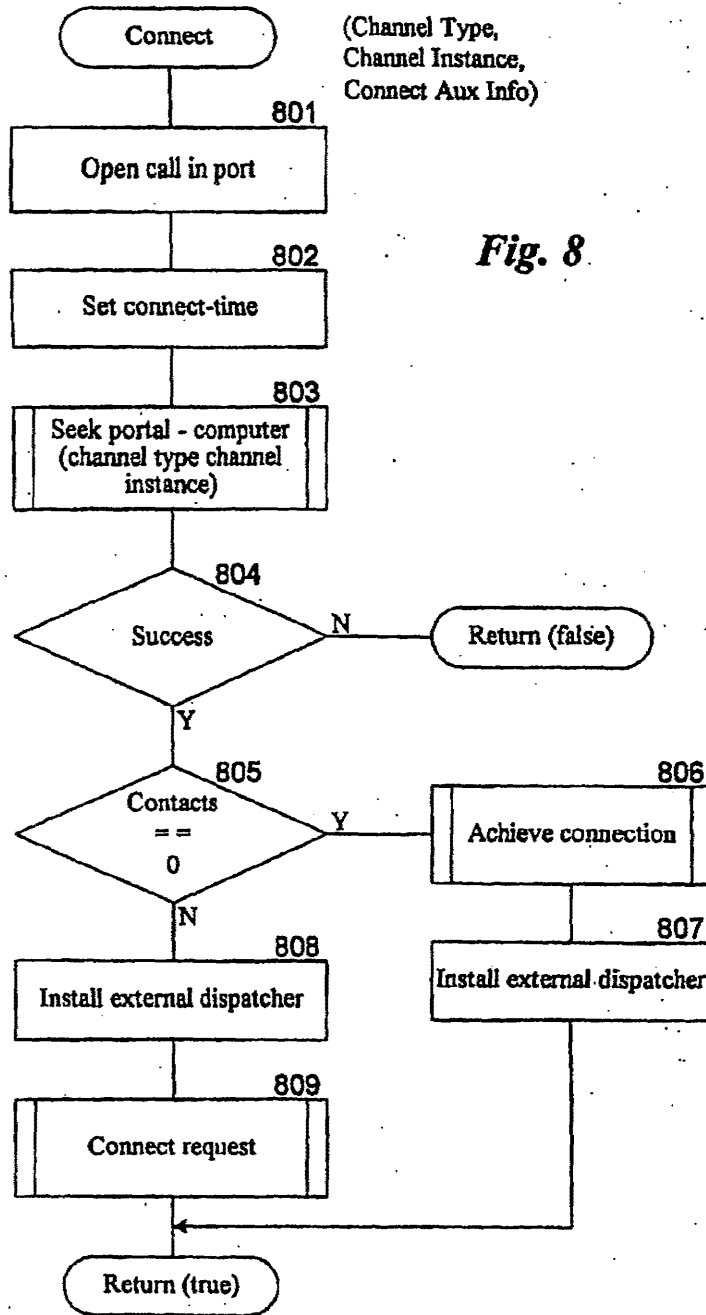


Fig. 7



REPLACEMENT SHEET





REPLACEMENT SHEET

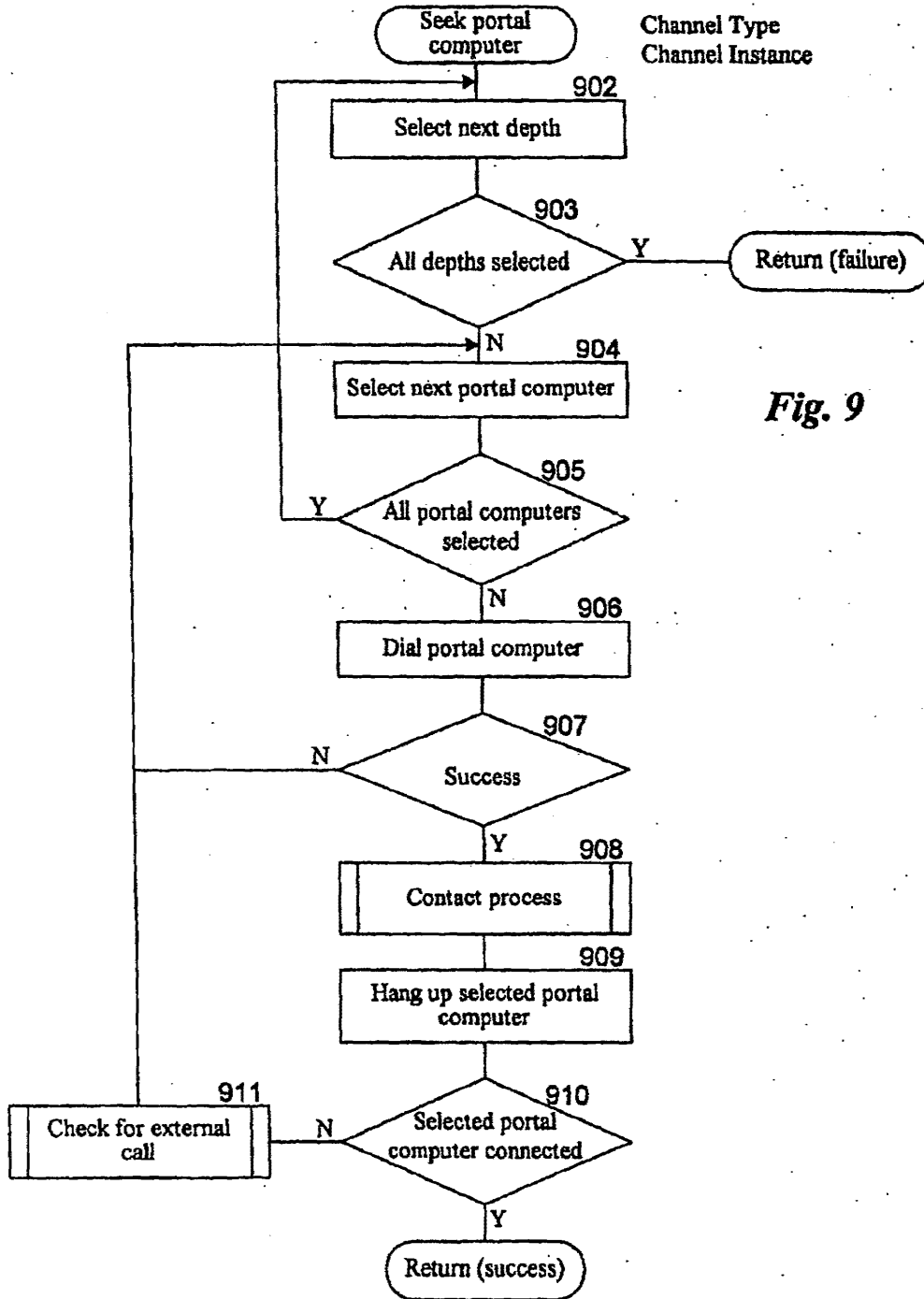


Fig. 9

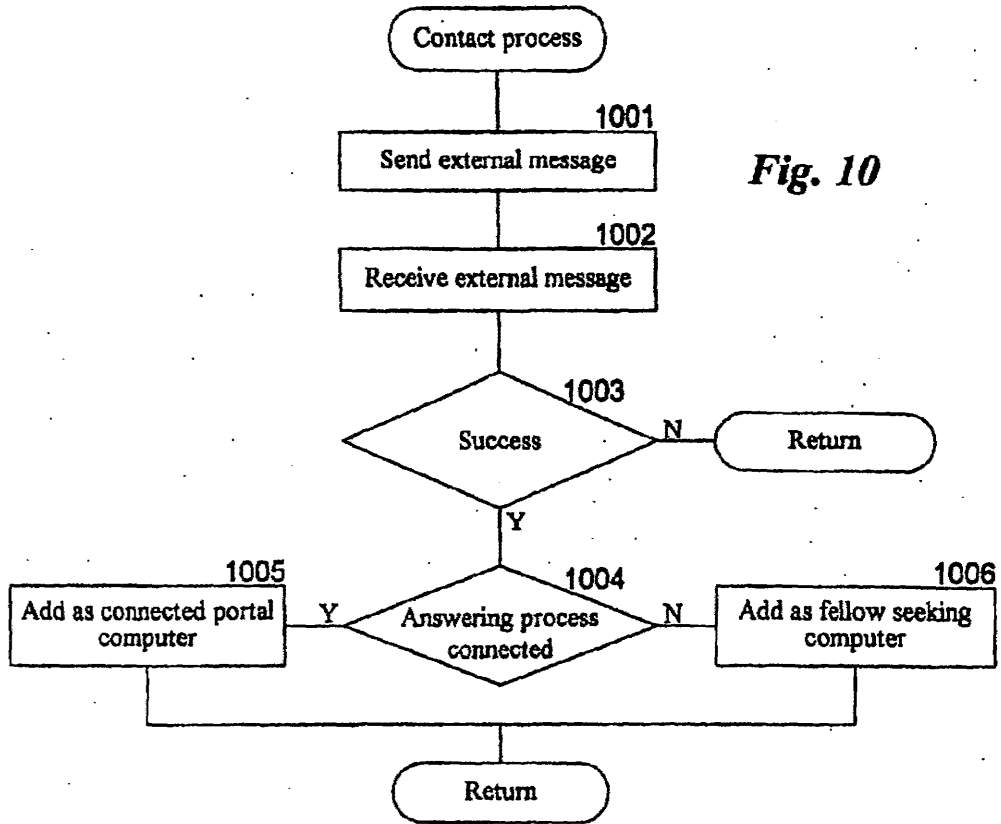
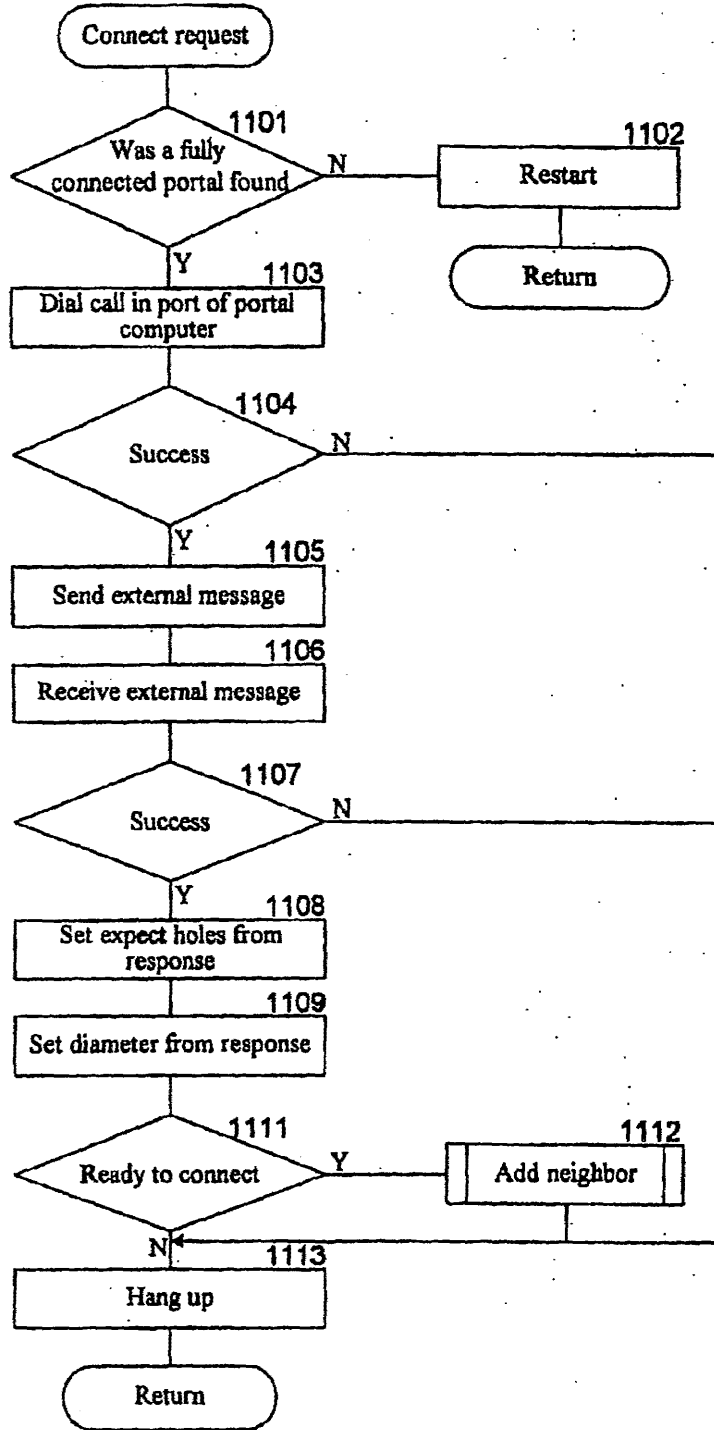


Fig. 10



REPLACEMENT SHEET

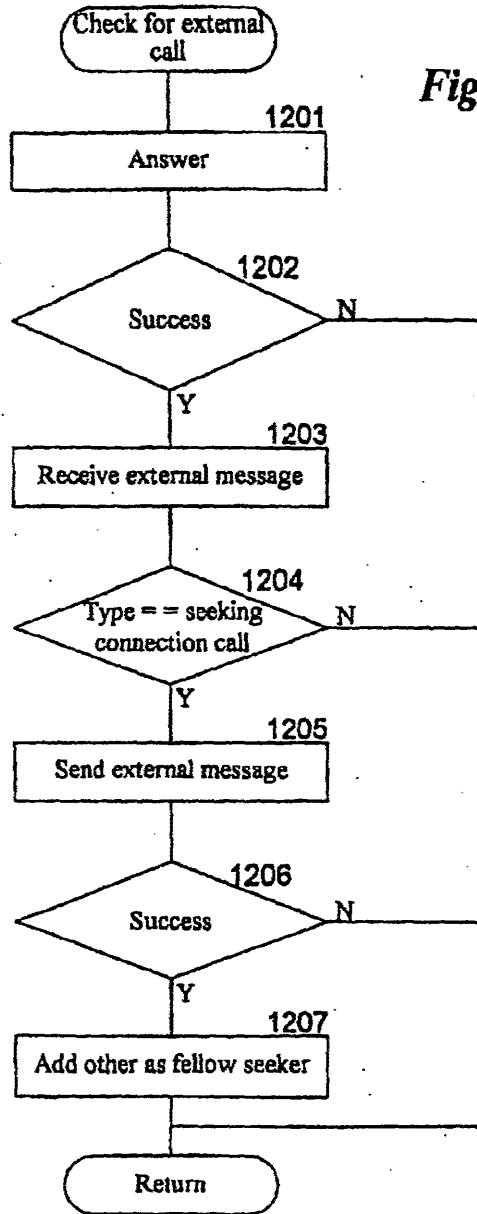
Fig. 11





REPLACEMENT SHEET

Fig. 12



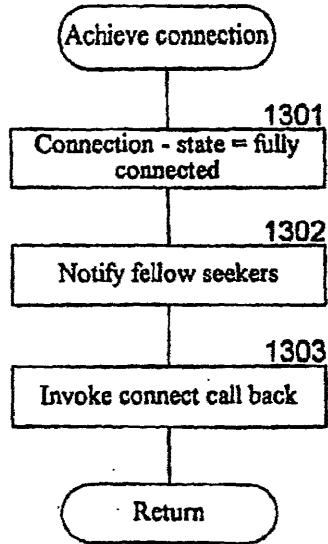
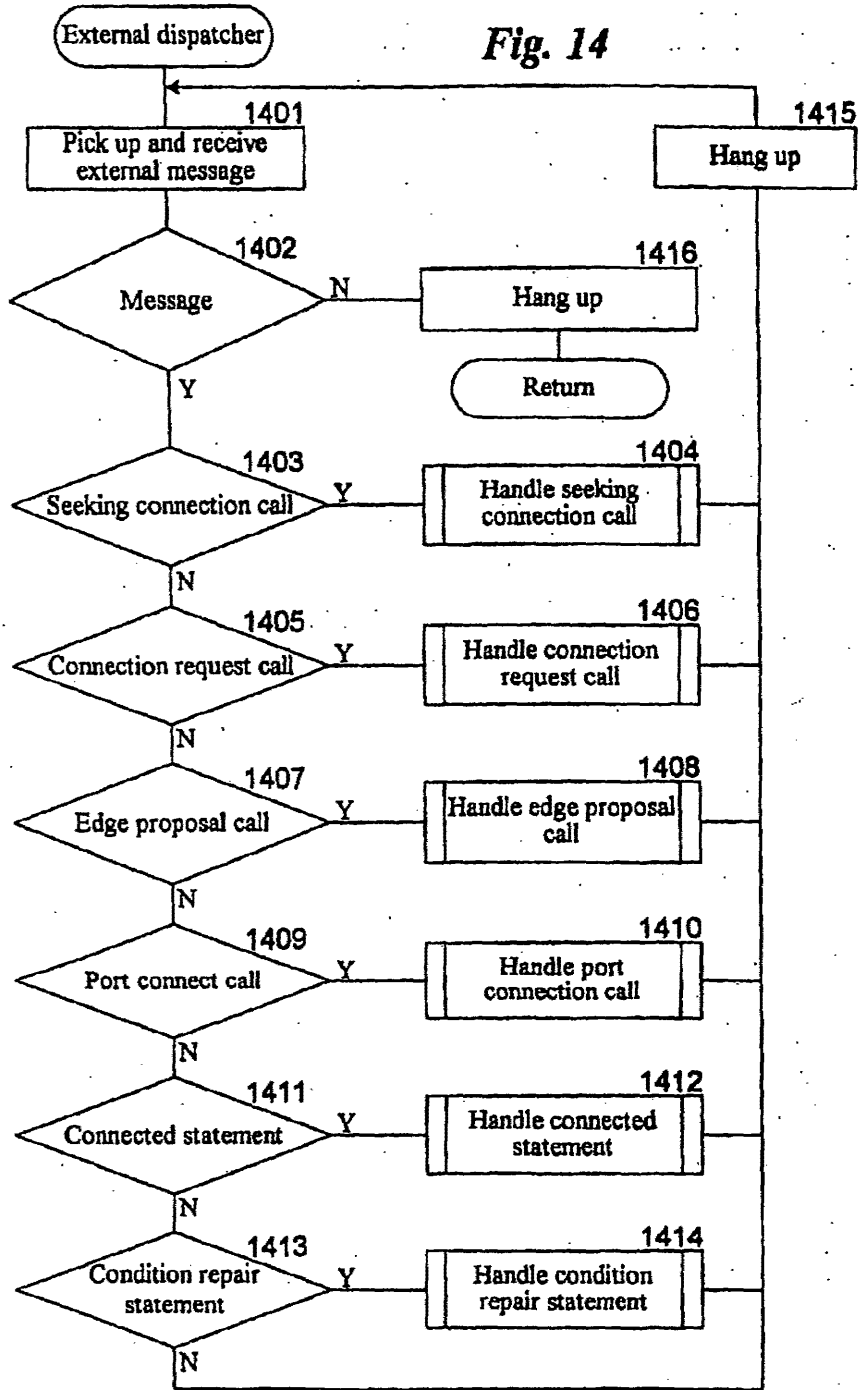


Fig. 13



REPLACEMENT SHEET

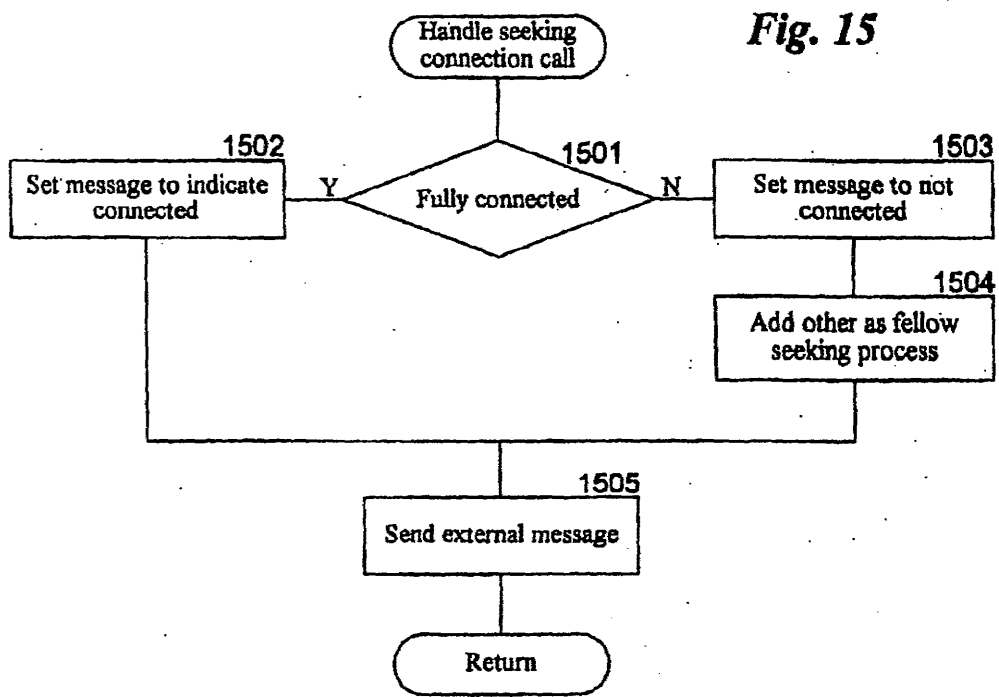
Fig. 14





REPLACEMENT SHEET

Fig. 15





REPLACEMENT SHEET

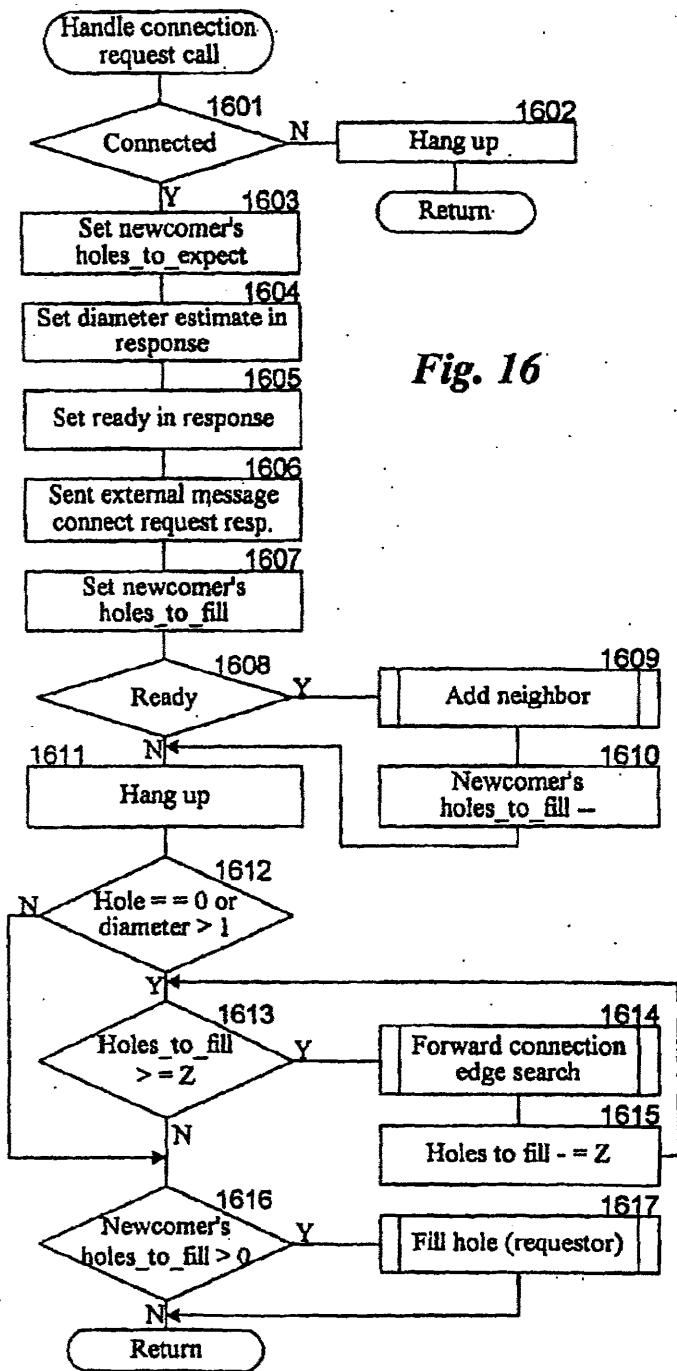


Fig. 16



Fig. 17

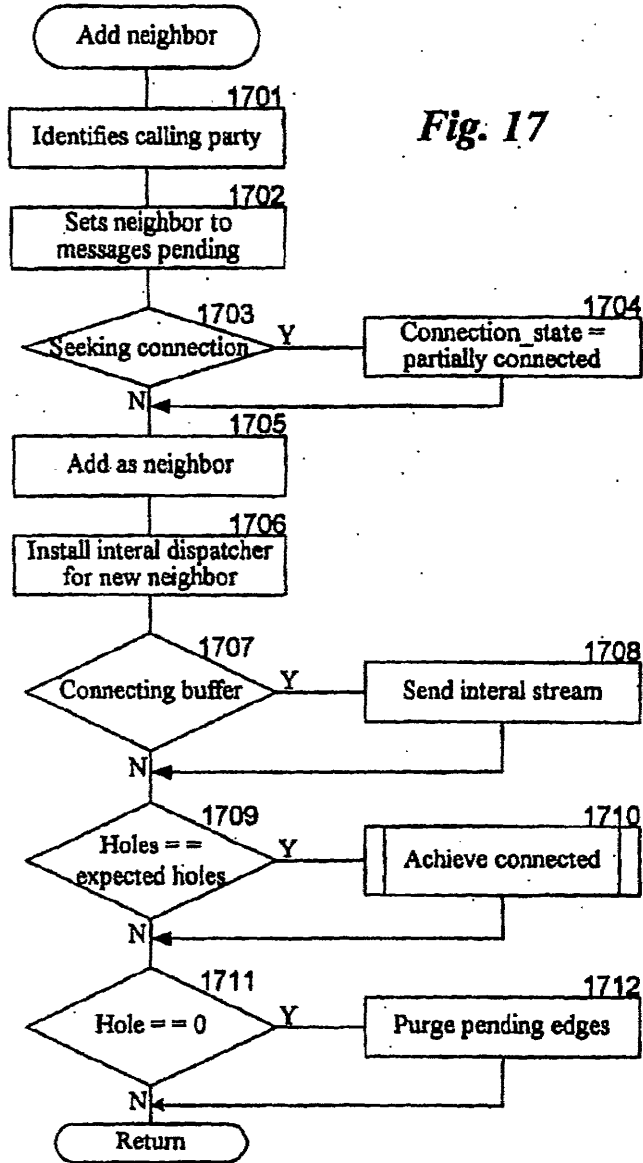




Fig. 18

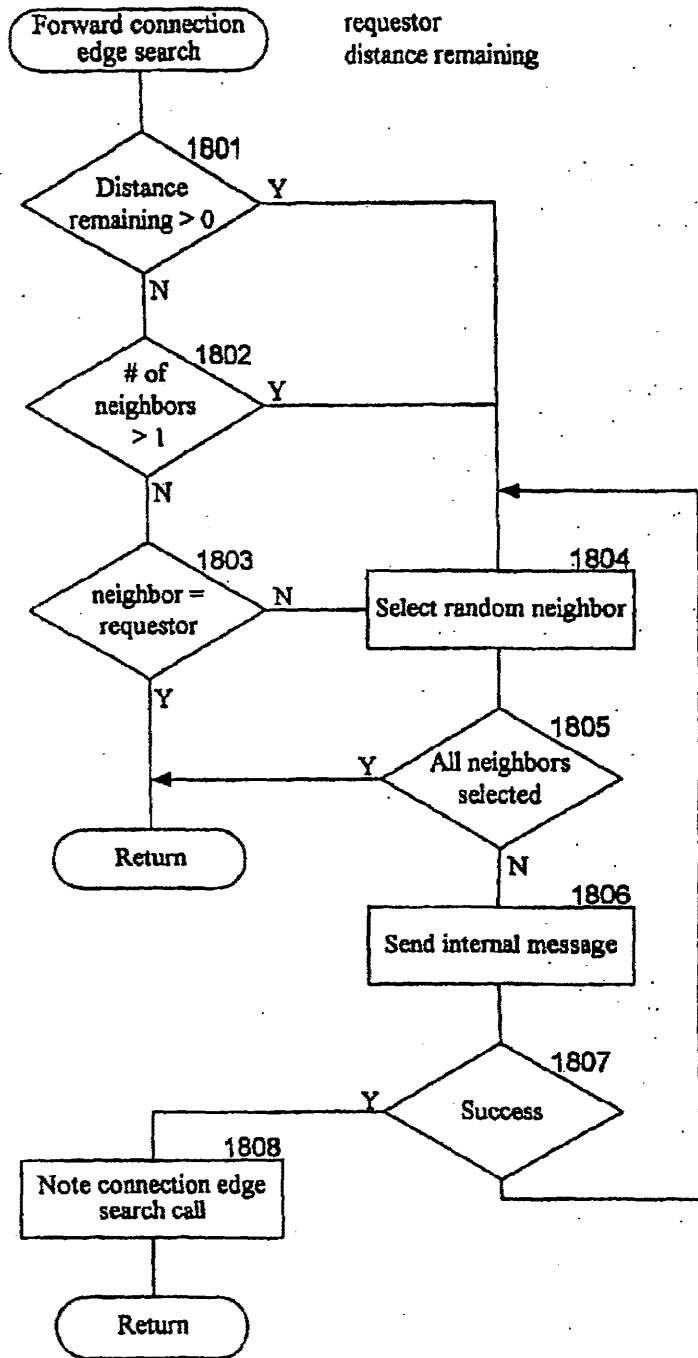




Fig. 20

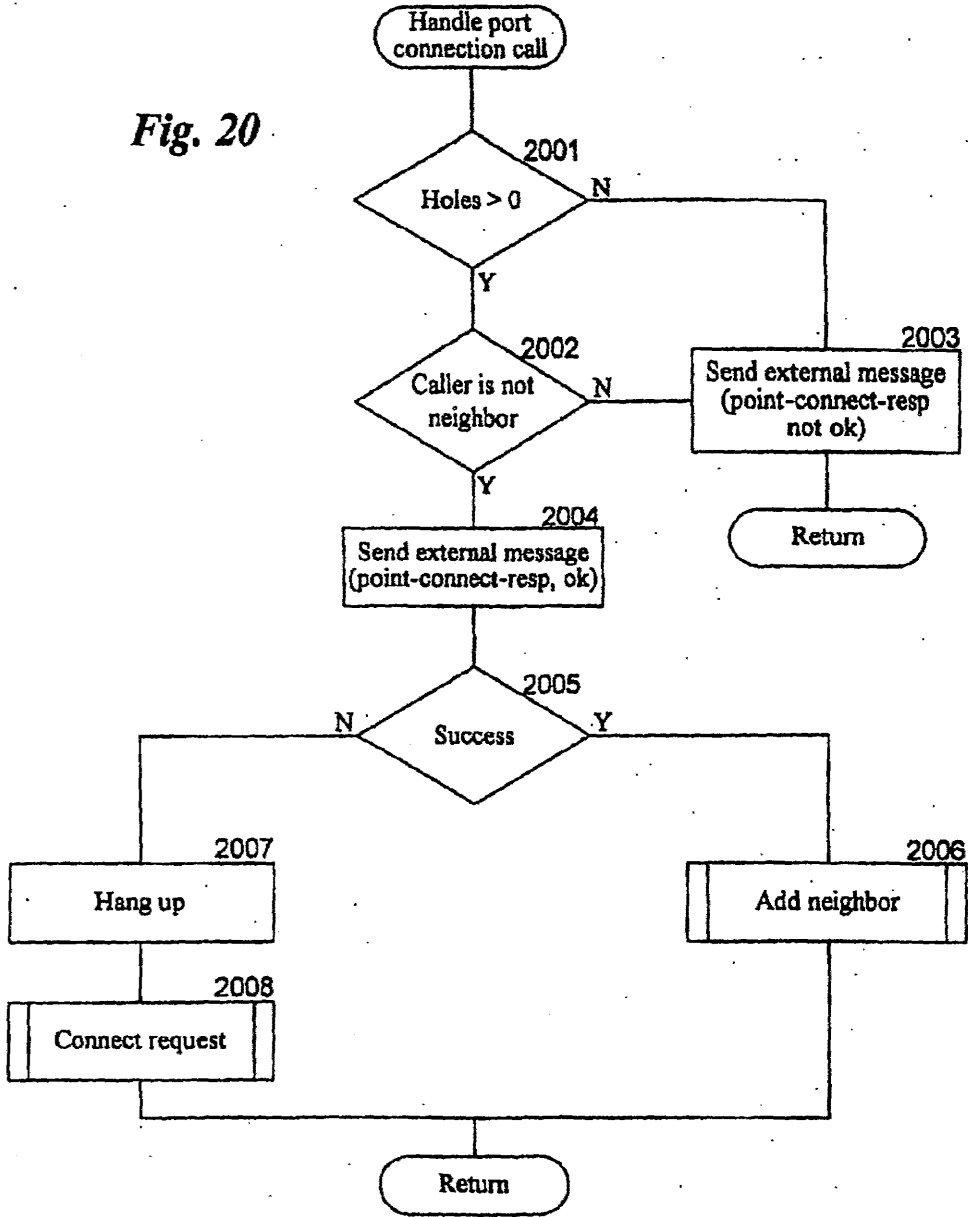
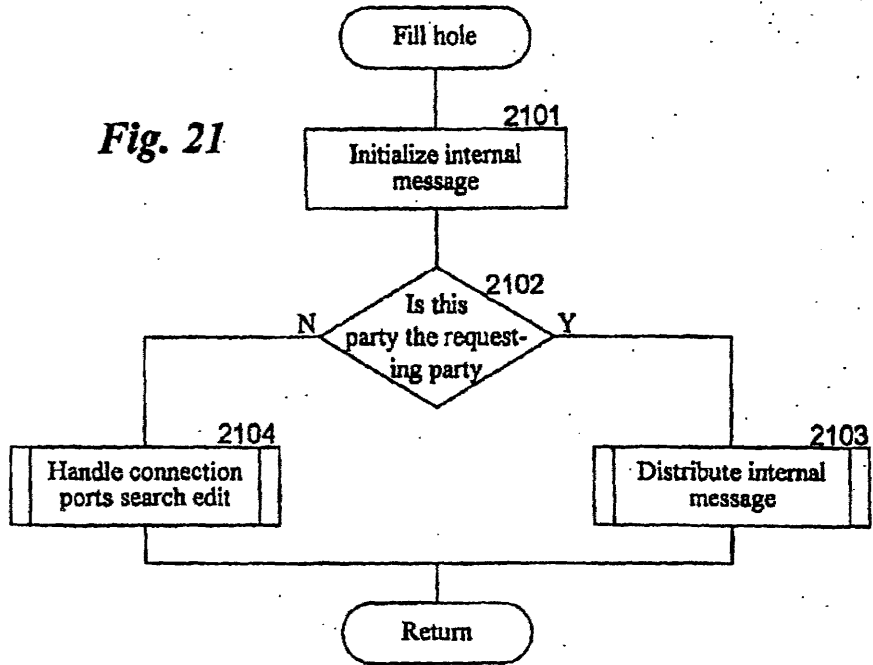




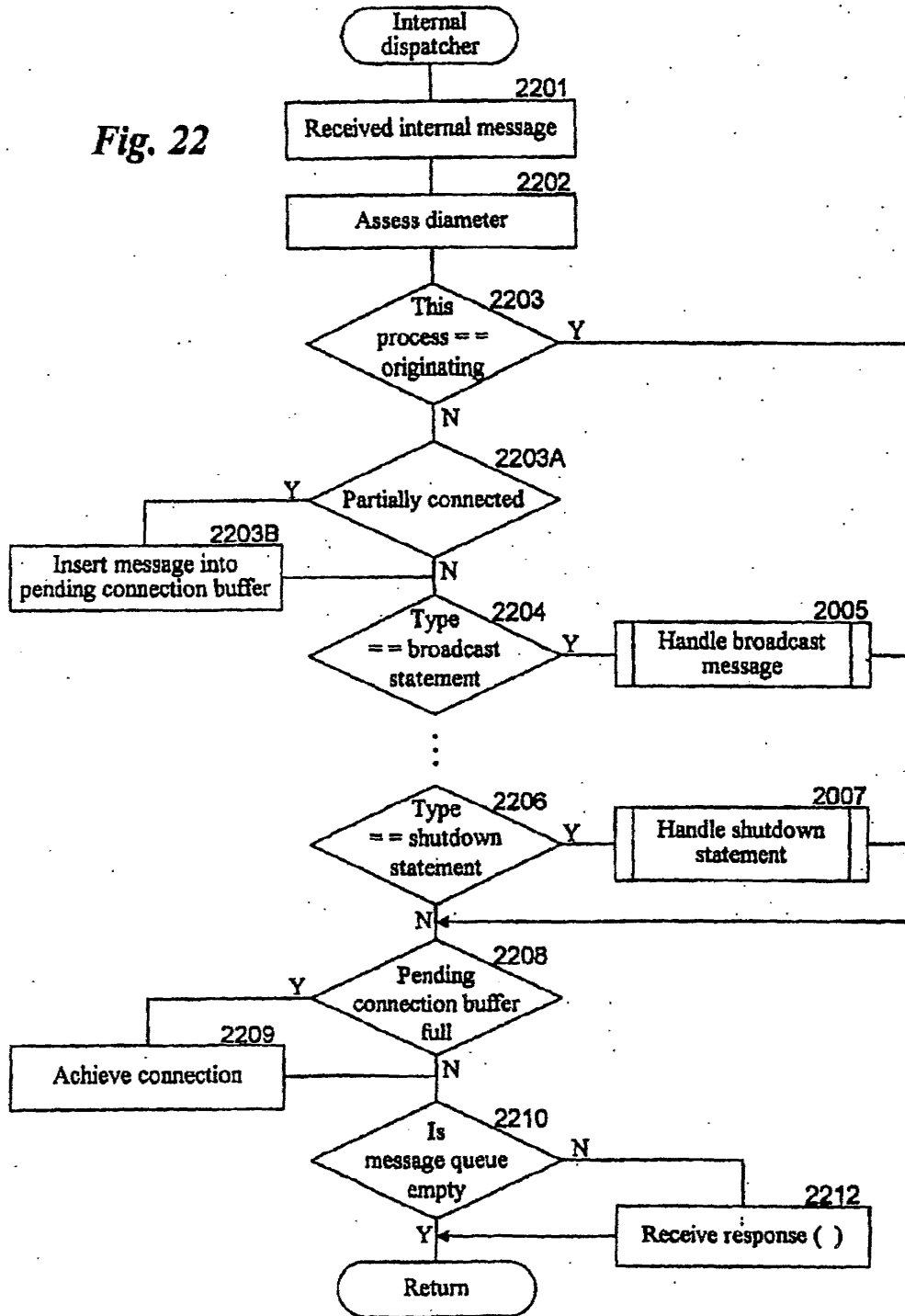
Fig. 21





REPLACEMENT SHEET

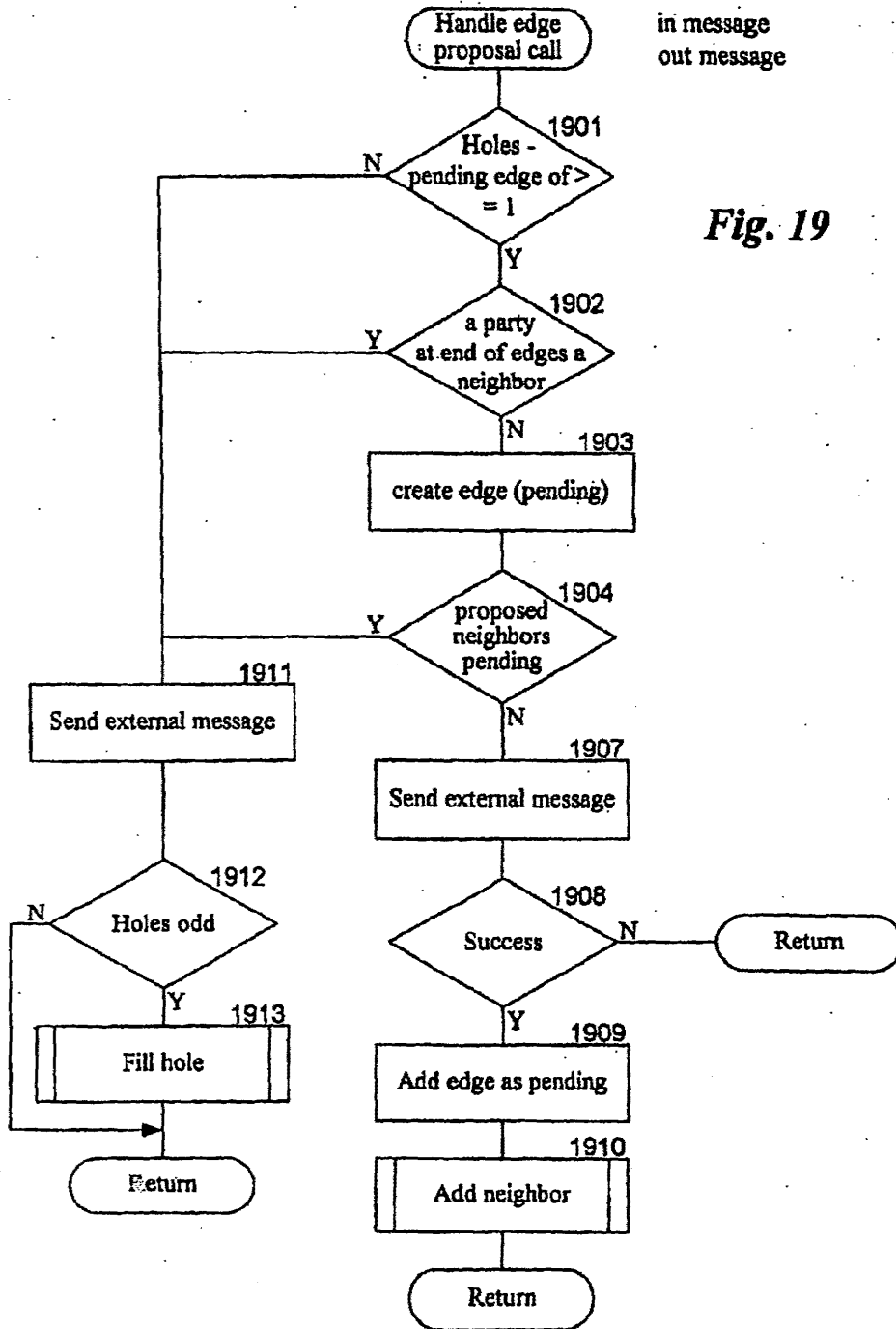
Fig. 22





in message
out message

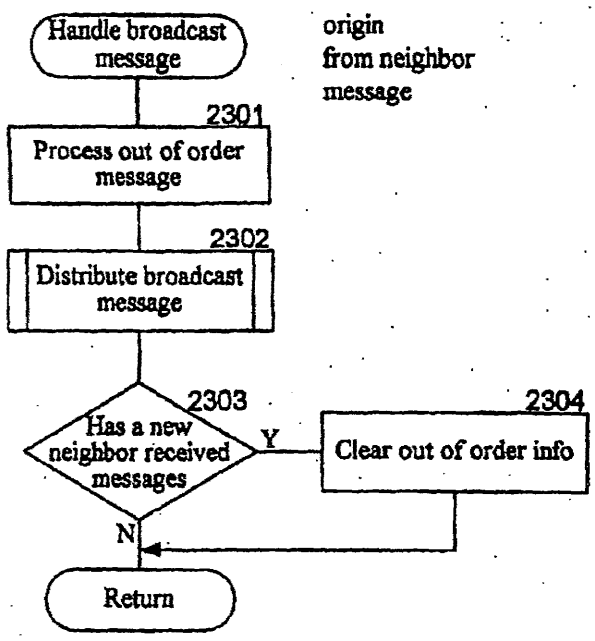
Fig. 19





REPLACEMENT SHEET

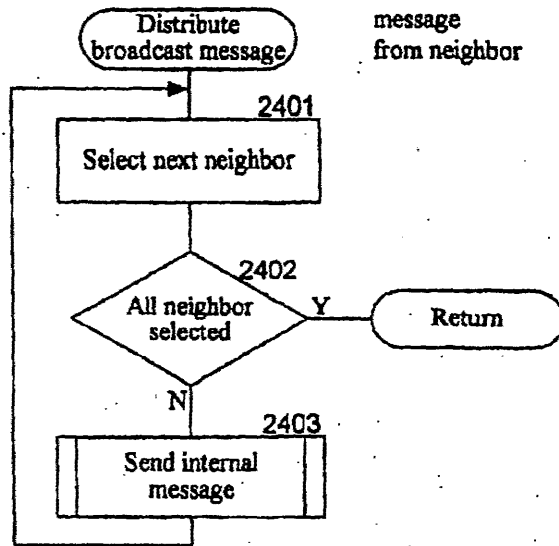
Fig. 23



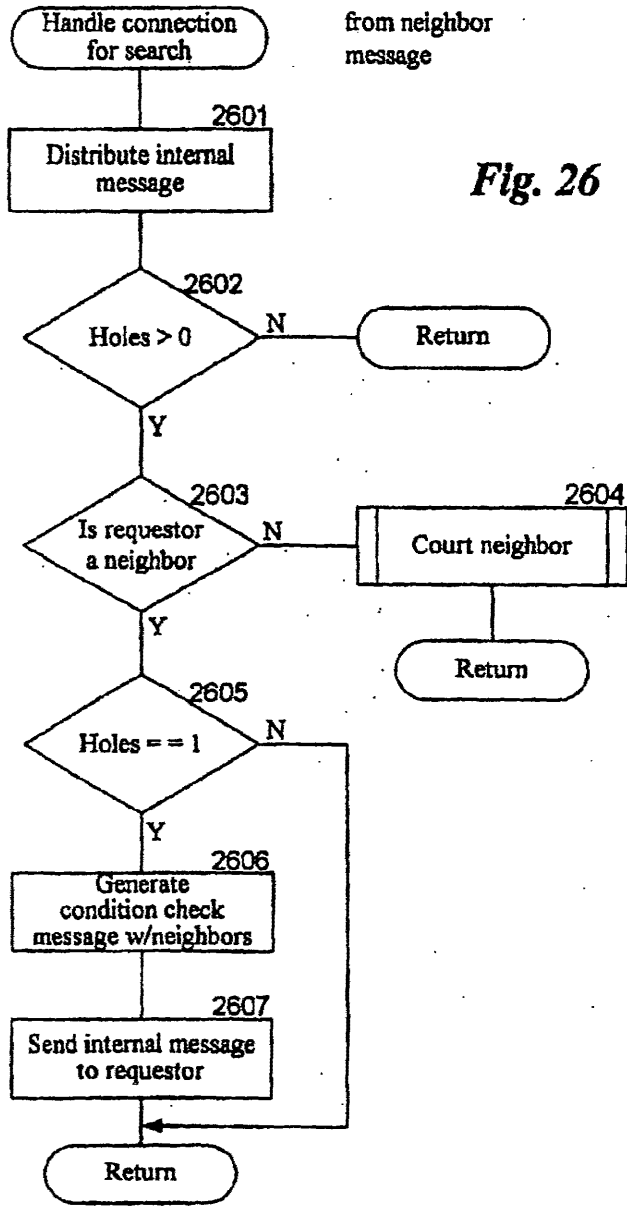


REPLACEMENT SHEET

Fig. 24



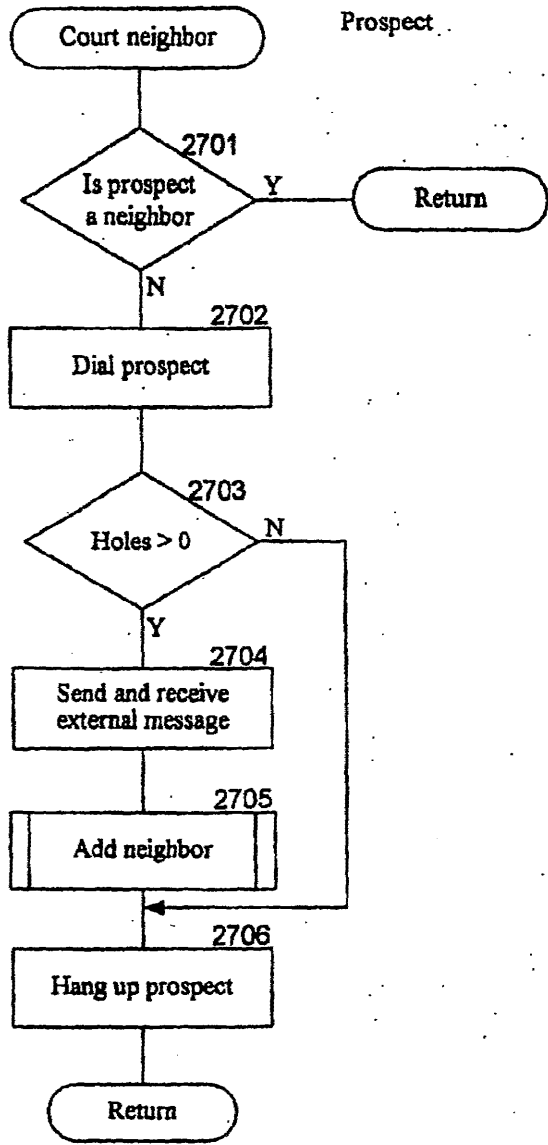
message from neighbor





REPLACEMENT SHEET

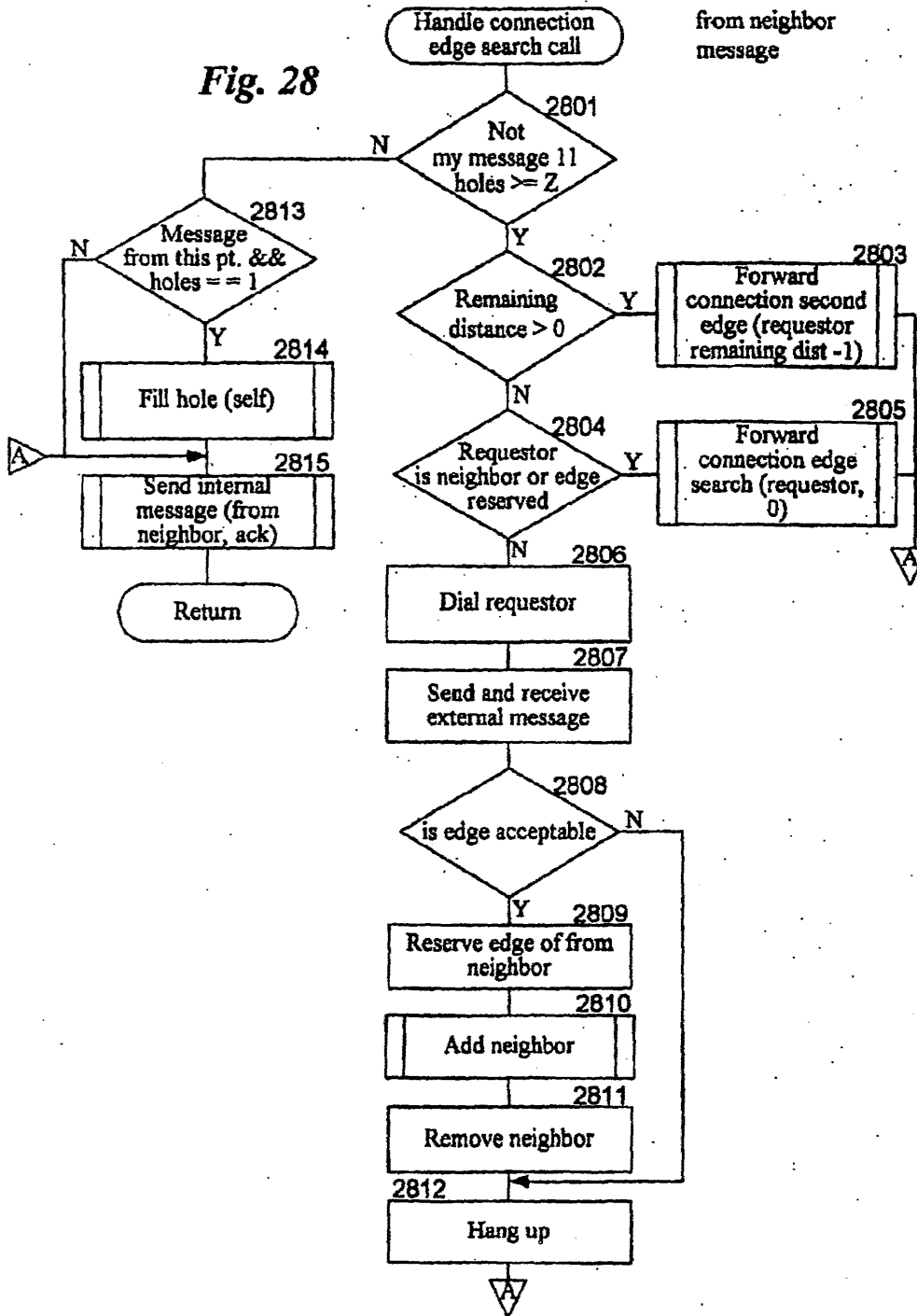
Fig. 27





REPLACEMENT SHEET

Fig. 28

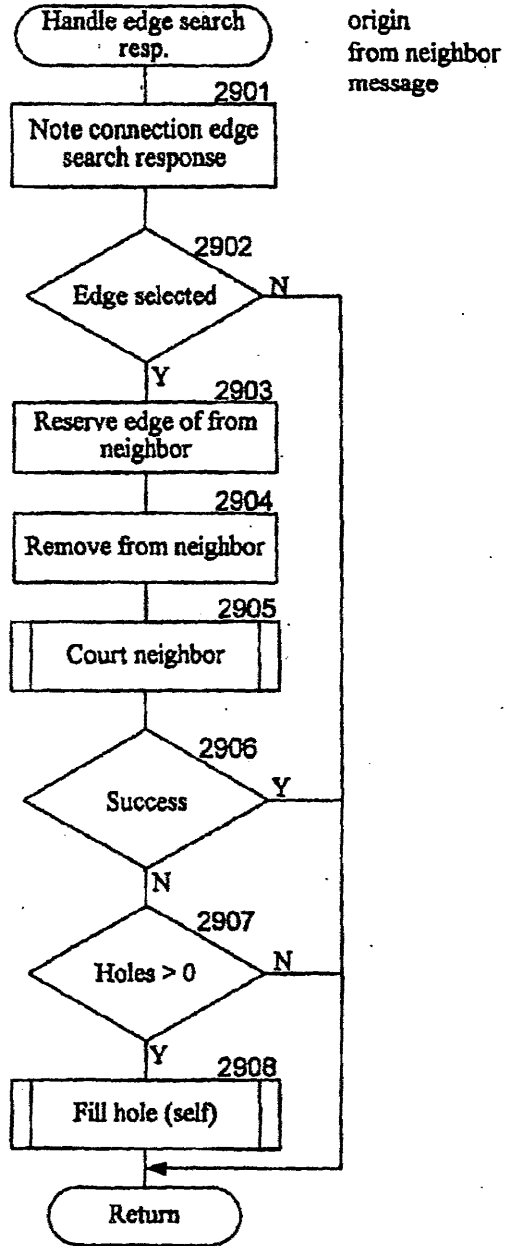


from neighbor message



REPLACEMENT SHEET

Fig. 29





REPLACEMENT SHEET

Fig. 30

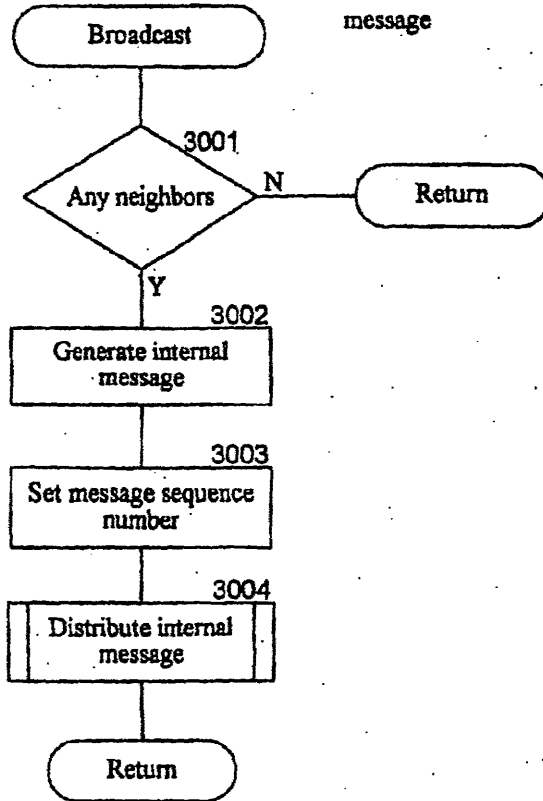




Fig. 31

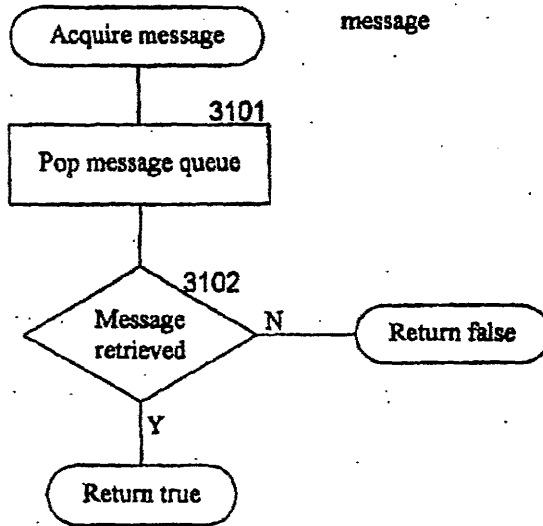
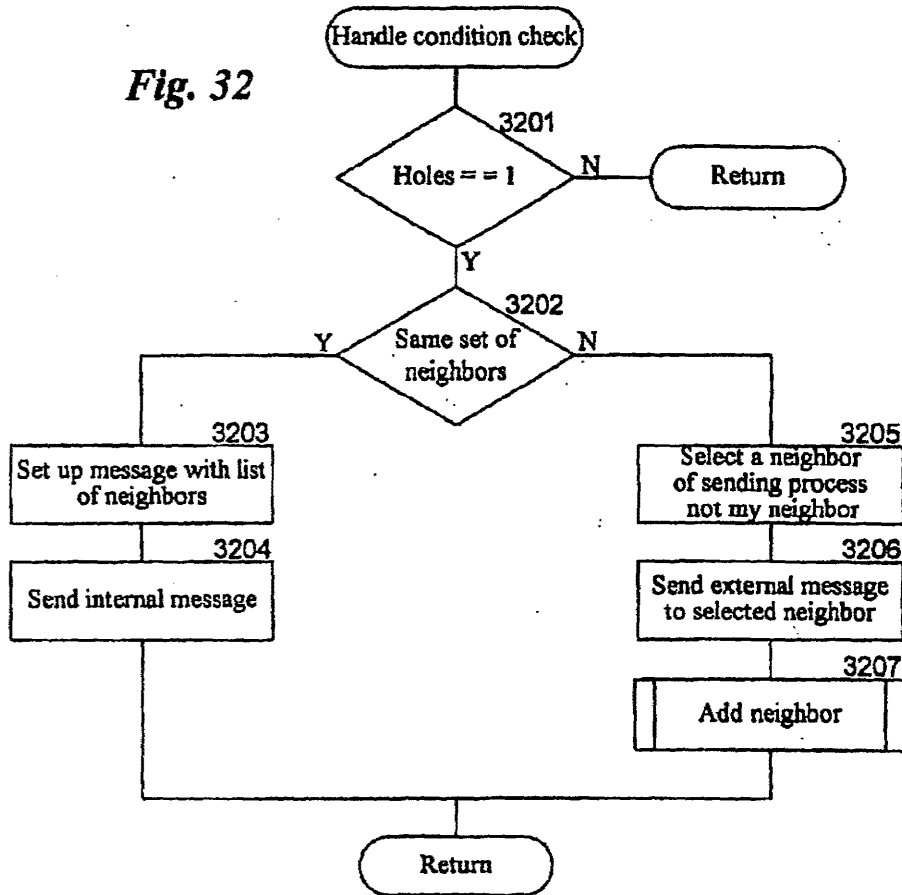




Fig. 32





REPLACEMENT SHEET

Fig. 33

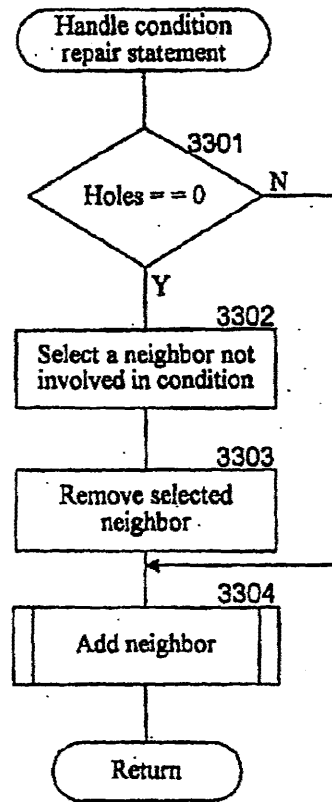
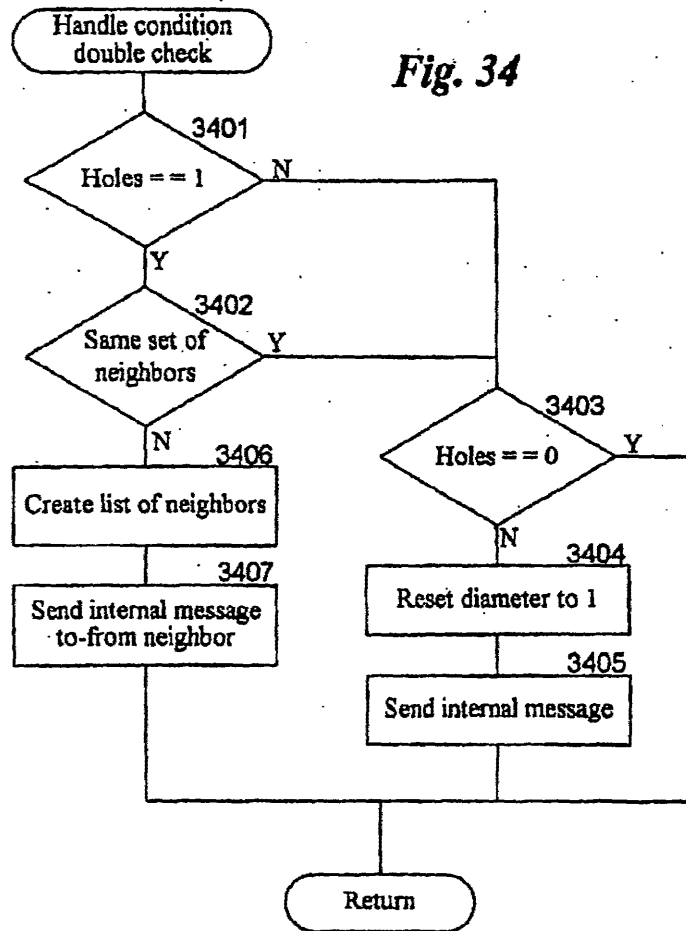




Fig. 34





US006829634B1

(12) **United States Patent**
Holt et al.

(10) **Patent No.:** **US 6,829,634 B1**
(45) **Date of Patent:** **Dec. 7, 2004**

- (54) **BROADCASTING NETWORK**
- (75) **Inventors:** **Fred B. Holt**, Seattle, WA (US); **Virgil E. Bourassa**, Bellevue, WA (US)
- (73) **Assignee:** **The Boeing Company**, Seattle, WA (US)
- (* **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 737 days.
- (21) **Appl. No.:** **09/629,576**
- (22) **Filed:** **Jul. 31, 2000**
- (51) **Int. Cl.⁷** **G06F 15/16**
- (52) **U.S. Cl.** **709/204; 709/205; 709/203; 709/243; 709/201; 709/238; 709/319; 709/225; 370/236**
- (58) **Field of Search** **709/106, 201, 709/238, 319**

- 5,864,711 A 1/1999 Mairs et al.
- 5,867,660 A 2/1999 Schmidt et al.
- 5,867,667 A 2/1999 Butman et al.
- 5,870,605 A 2/1999 Bracho et al.
- 5,874,960 A 2/1999 Mairs et al.
- 5,899,980 A 5/1999 Wilf et al.
- 5,907,610 A 5/1999 Onweller
- 5,928,335 A 7/1999 Morita
- 5,935,215 A 8/1999 Bell et al.
- 5,948,054 A 9/1999 Nielsen
- 5,949,975 A 9/1999 Batty et al.
- 5,953,318 A * 9/1999 Natikemper et al. 370/236
- 5,956,484 A 9/1999 Rosenberg et al.
- 5,974,043 A 10/1999 Solomon
- 5,987,506 A 11/1999 Carter et al.

(List continued on next page.)

OTHER PUBLICATIONS

Alagar, S. and Venkatesan, S., "Reliable Broadcast in Mobile Wireless Networks," Department of Computer Science, University of Texas at Dallas, Military Communications Conference, 1995, MILCOM '95 Conference Record, IEEE San Diego, California, Nov. 5-8, 1995 (pp. 236-240).

(List continued on next page.)

(56) **References Cited**

U.S. PATENT DOCUMENTS

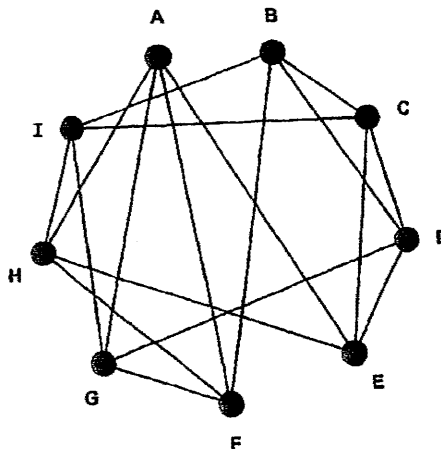
- 4,912,656 A 3/1990 Cain et al.
- 5,056,085 A 10/1991 Vu
- 5,309,437 A 5/1994 Perlman et al.
- 5,426,637 A 6/1995 Derby et al.
- 5,535,199 A 7/1996 Amri et al.
- 5,568,487 A 10/1996 Sitbon et al.
- 5,636,371 A 6/1997 Yu
- 5,673,265 A 9/1997 Gupta et al.
- 5,696,903 A 12/1997 Mahany
- 5,732,074 A 3/1998 Spaur et al.
- 5,732,219 A 3/1998 Blumer et al.
- 5,734,865 A 3/1998 Yu
- 5,737,526 A 4/1998 Periasamy et al.
- 5,754,830 A 5/1998 Butts et al.
- 5,761,425 A 6/1998 Miller
- 5,764,756 A 6/1998 Onweller
- 5,790,548 A 8/1998 Sistanizadeh et al.
- 5,790,553 A 8/1998 Deaton, Jr. et al.
- 5,799,016 A 8/1998 Onweller
- 5,802,285 A 9/1998 Hirviniemi

Primary Examiner—Hosain Alam
Assistant Examiner—Young N. Won
(74) *Attorney, Agent, or Firm*—Perkins Coie LLP

(57) **ABSTRACT**

A technique for broadcasting data across a network is provided. An originating participant sends data to another participant, which in turn sends the data that it receives from a neighbor participant to its other neighbor participants. Communication in the broadcast network is controlled by a contact module that locates the neighbor participants to which the seeking participant can be connected and by a join module that establishes the connection between the neighbor participants and the seeking participant. Data is numbered sequentially so that data that is received out of order can be queued and rearranged.

24 Claims, 39 Drawing Sheets



U.S. PATENT DOCUMENTS

6,003,088 A	12/1999	Houston et al.	
6,013,107 A	1/2000	Blackshear et al.	
6,023,734 A	2/2000	Ratcliff et al.	
6,029,171 A	2/2000	Smiga et al.	
6,032,188 A	2/2000	Mairs et al.	
6,038,602 A	3/2000	Ishikawa	
6,047,289 A	4/2000	Thorne et al.	
6,094,676 A	7/2000	Gray et al.	
6,199,116 B1	3/2001	May et al.	
6,216,177 B1	4/2001	Mairs et al.	
6,223,212 B1	4/2001	Batty et al.	
6,243,691 B1	6/2001	Fisher et al.	
6,268,855 B1	7/2001	Mairs et al.	
6,271,839 B1	8/2001	Mairs et al.	
6,285,363 B1	9/2001	Mairs et al.	
6,304,928 B1	10/2001	Mairs et al.	
6,611,872 B1 *	8/2003	McCanne	709/238

OTHER PUBLICATIONS

International Search Report for The Boeing Company, International Patent Application No. PCT/US01/24240, Jun. 5, 2002 (7 pages).

U.S. patent application Ser. No. 09/629,570, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,577, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,575, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,572, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,023, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,043, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,024, Bourassa et al., filed Jul. 31, 2000.

U.S. patent application Ser. No. 09/629,042, Bourassa et al., filed Jul. 31, 2000.

Murphy, Patricia, A., "The Next Generation Networking Paradigm: Producer/Consumer Model," *Dedicated Systems Magazine*—2000 (pp. 26–28).

The Gamer's Guide, "First-Person Shooters," Oct. 20, 1998 (4 pages).

The O'Reilly Network, "Gnutella: Alive, Well, and Changing Fast," Jan. 25, 2001 (5 pages) <http://www.open2p.com/lpt/> . . . [Accessed Jan. 29, 2002].

Oram, Andy, "Gnutella and Freenet Represents True Technological Innovation," May 12, 2000 (7 pages) The O'Reilly Network <http://www.oreillynet.com/lpt> . . . [Accessed Jan. 29, 2002].

Internetworking Technologies Handbook, Chapter 43 (pp. 43–1–43–16).

Oram, Andy, "Peer-to-Peer Makes the Internet Interesting Again," Sep. 22, 2000 (7 pages) The O'Reilly Network <http://linux.oreillynet.com/lpt> . . . [Accessed Jan. 29, 2002].

Monte, Richard, "The Random Walk for Dummies," *MIT Undergraduate Journal of Mathematics* (pp. 143–148).

Srinivasan, R., "XDR: External Data Representation Standard," Sun Microsystems, Aug. 1995 (20 pages) Internet RFC/STD/FYI/BCP Archives <http://www.faqs.org/rfcs/rfc1832.html> [Accessed Jan. 29, 2002].

A Databeam Corporate White Paper, "A Primer on the T.120 Series Standards," Copyright 1995 (pp. 1–16).

Kessler, Gary, C., "An Overview of TCP/IP Protocols and the Internet," Apr. 23, 1999 (23 pages) Hill Associates, Inc. <http://www.hill.com/library/publications/t> . . . [Accessed Jan. 29, 2002].

Bondy, J.A., and Murty, U.S.R., "Graph Theory with Applications," Chapters 1–3 (pp. 1–47), 1976 American Elsevier Publishing Co., Inc., New York, New York.

Cormen, Thomas H. et al., *Introduction to Algorithms*, Chapter 5.3 (pp. 84–91), Chapter 12 (pp. 218–243), Chapter 13 (p. 245), 1990, The MIT Press, Cambridge, Massachusetts, McGraw-Hill Book Company, New York.

The Common Object Request Broker: Architecture and Specification, Revision 2.6, Dec. 2001, Chapter 12 (pp. 12–1–12–10), Chapter 13 (pp. 13–1–13–56), Chapter 16 (pp. 16–1–16–26), Chapter 18 (pp. 18–1–18–52), Chapter 20 (pp. 20–1–20–22).

The University of Warwick, Computer Science Open Days, "Demonstration on the Problems of Distributed Systems," <http://www.dcs.warwick.ac.u> . . . [Accessed Jan. 29, 2002].

* cited by examiner

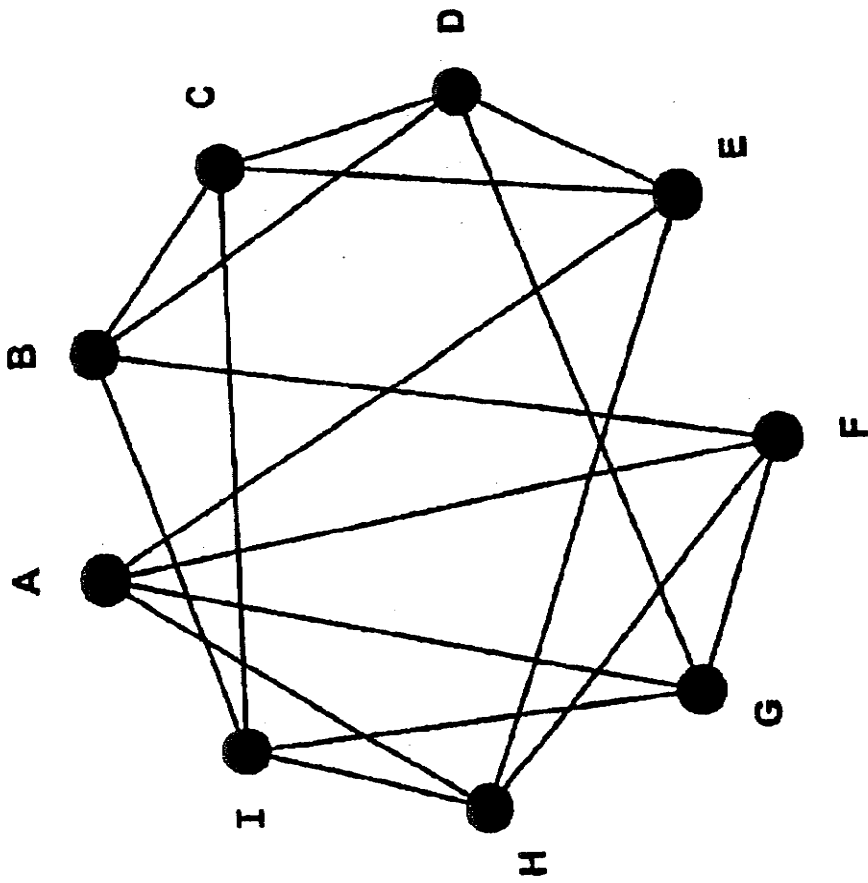


Fig. 1

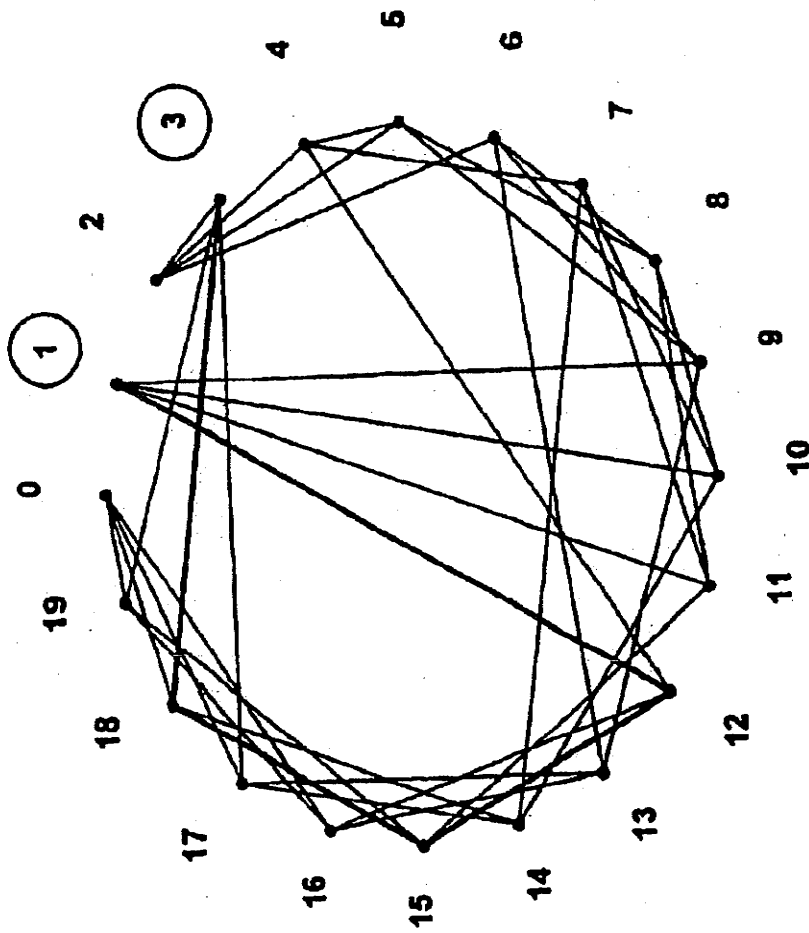


Fig. 2

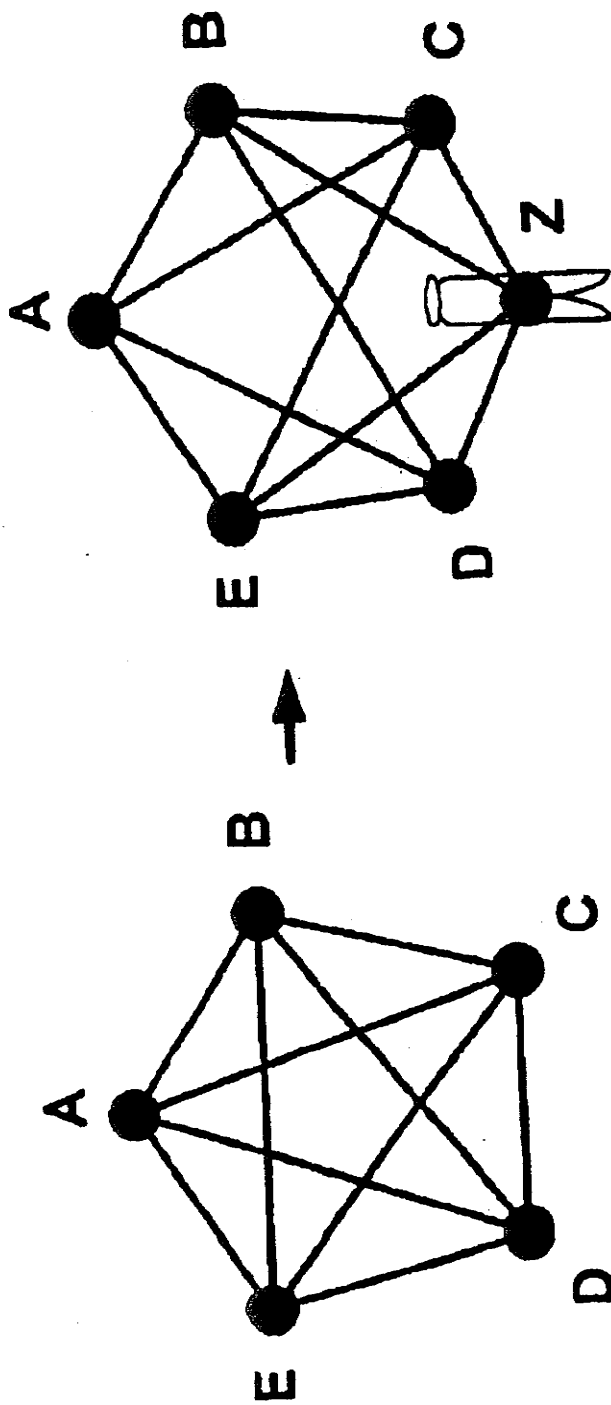


Fig. 3B

Fig. 3A

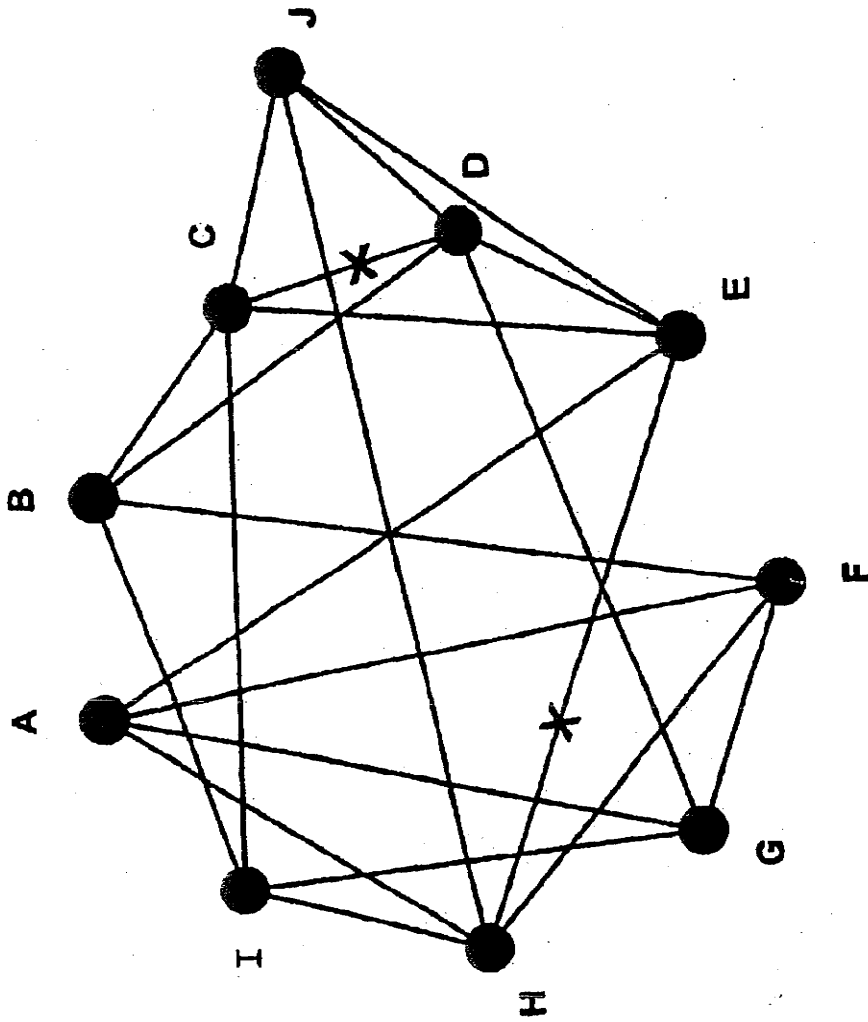


Fig. 4A

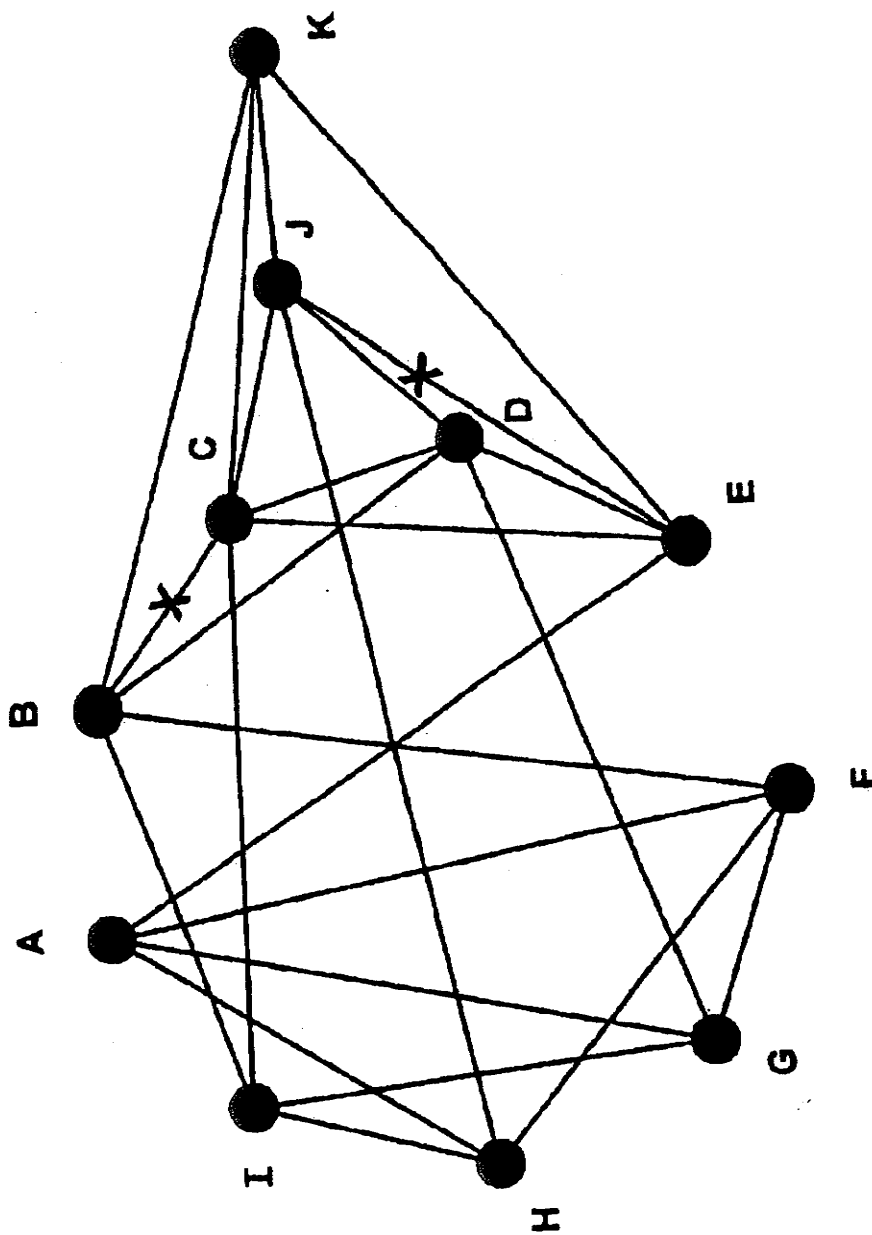


Fig. 4B

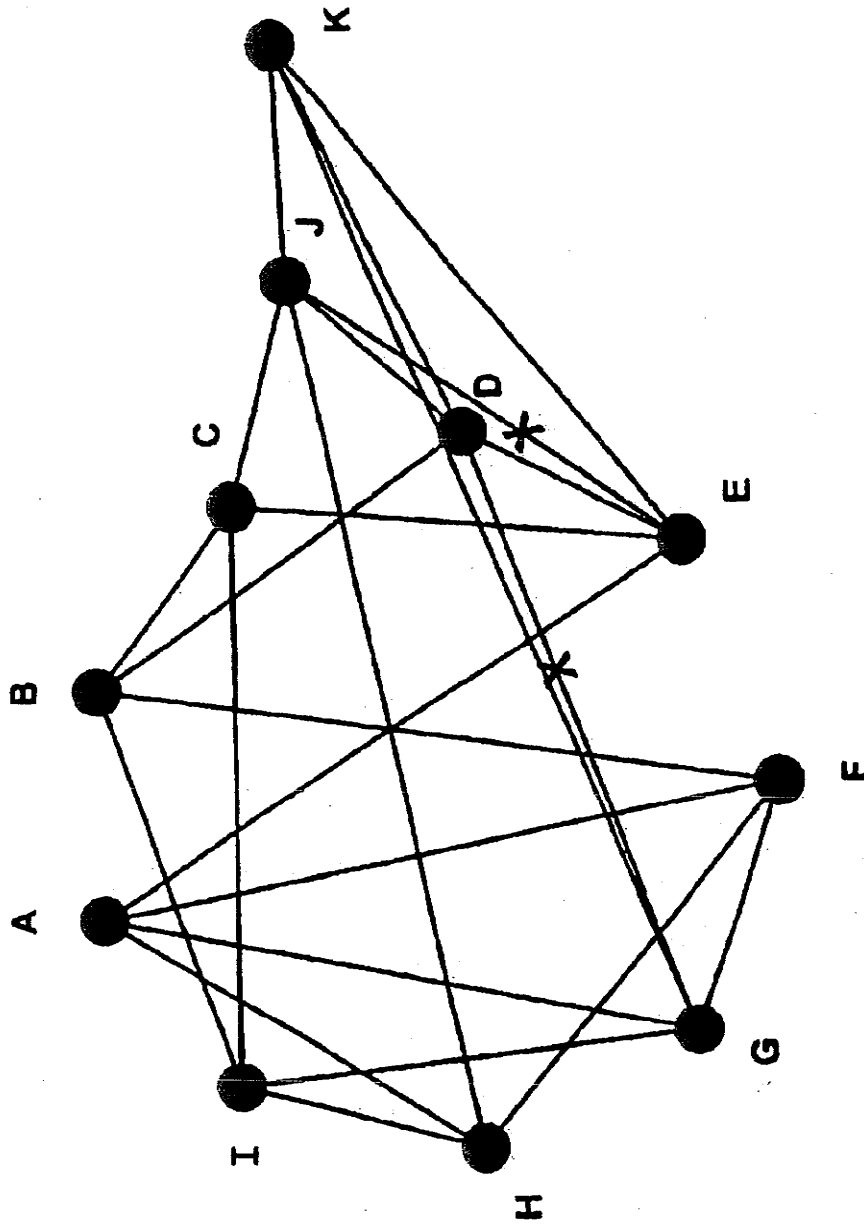


Fig. 4C

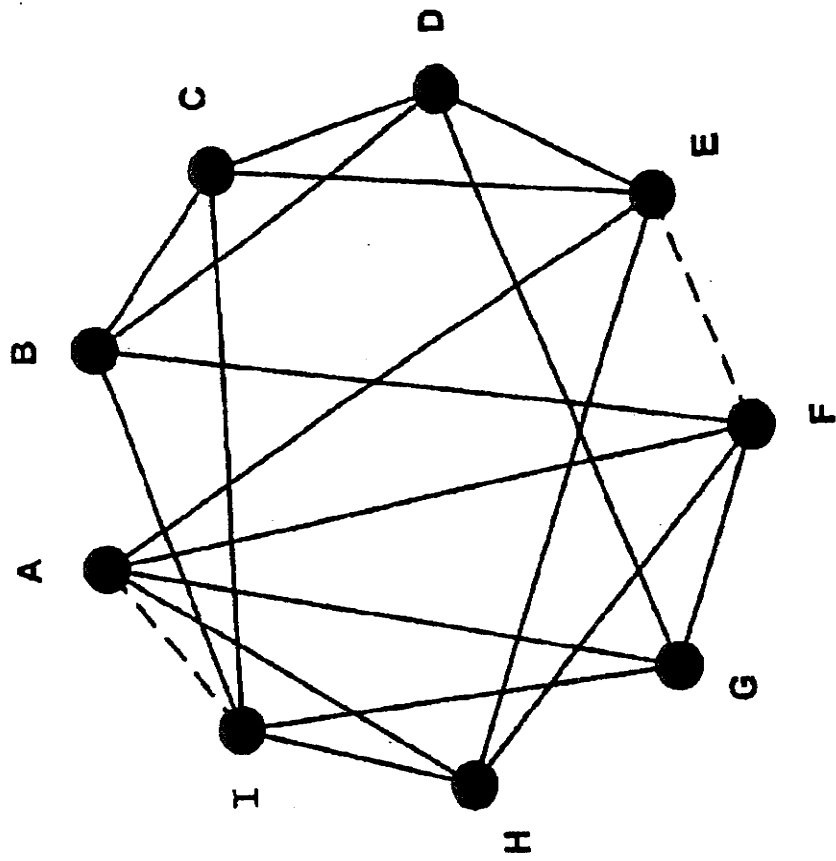


Fig. 5A

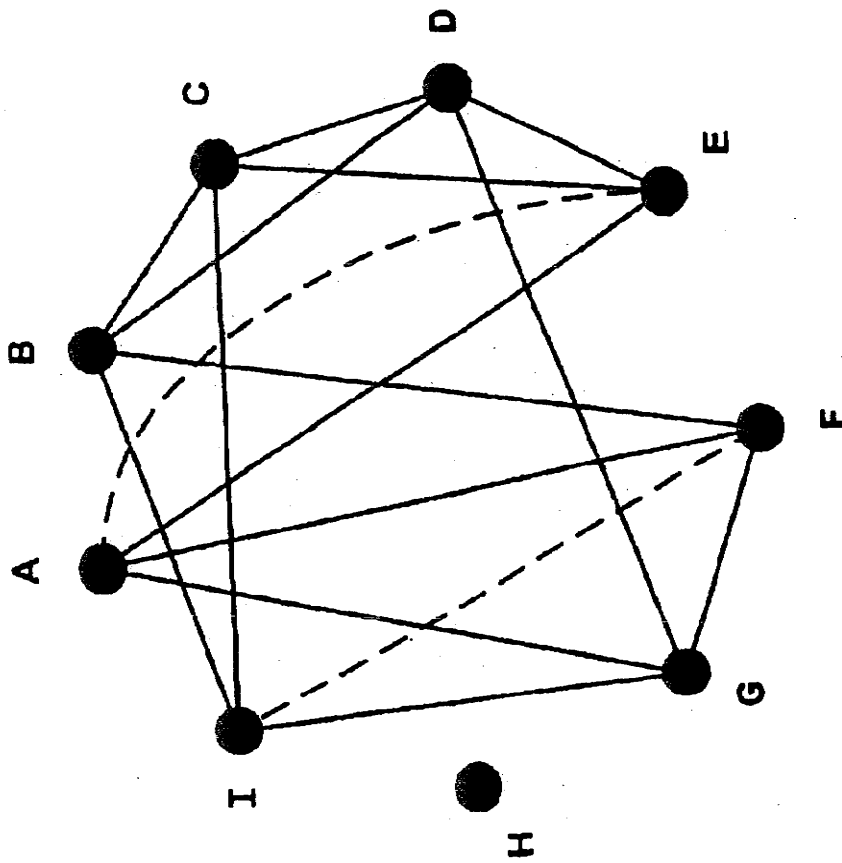


Fig. 5B

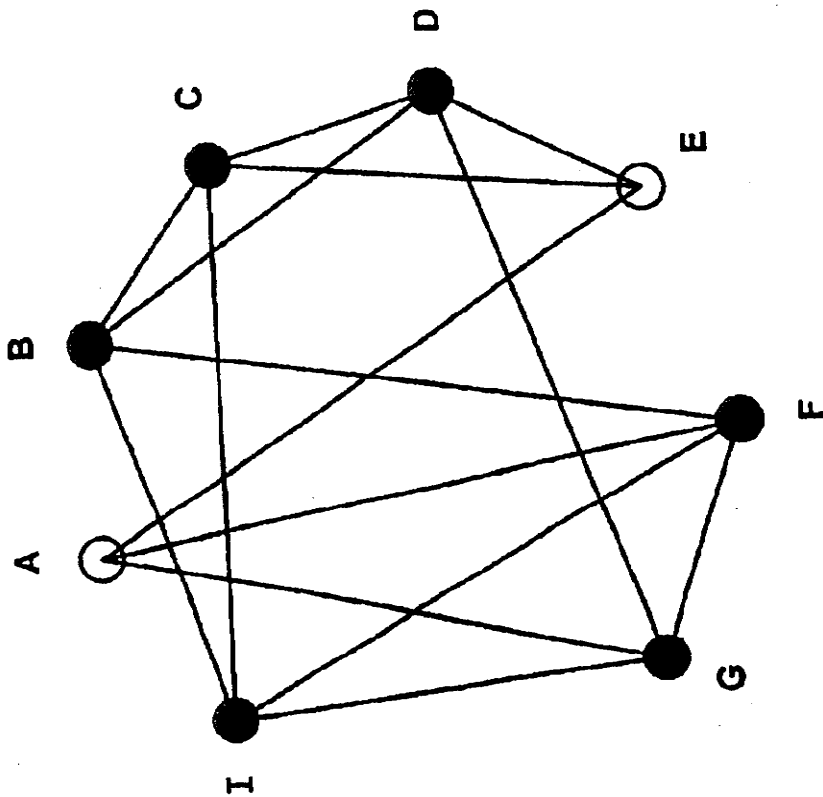


Fig. 5C

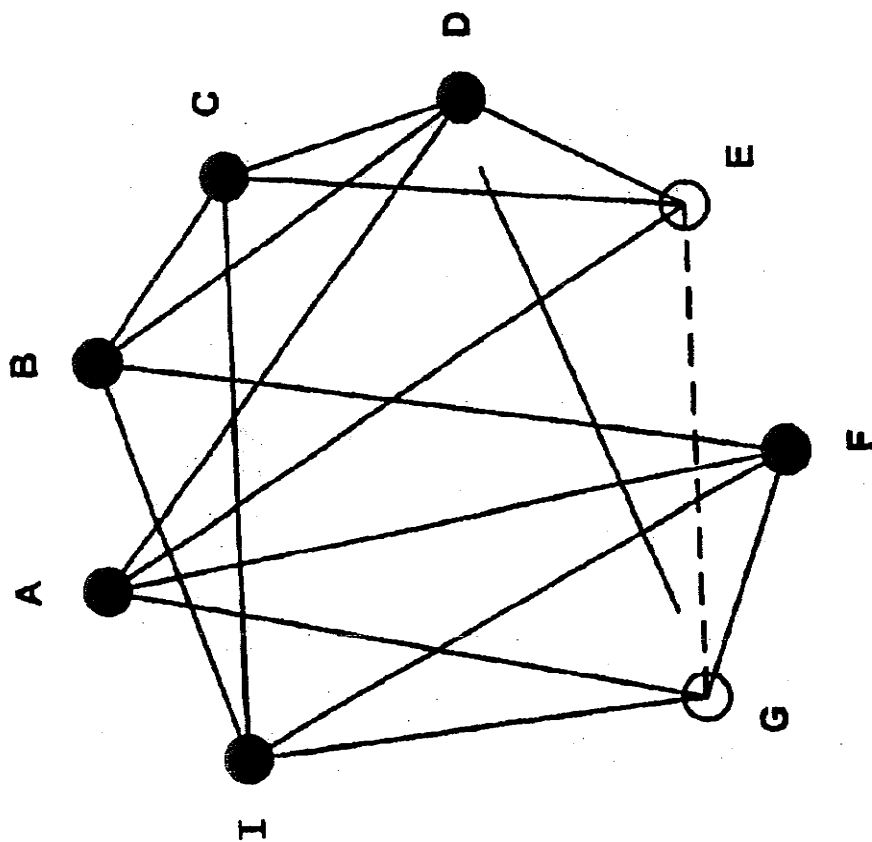


Fig. 5D

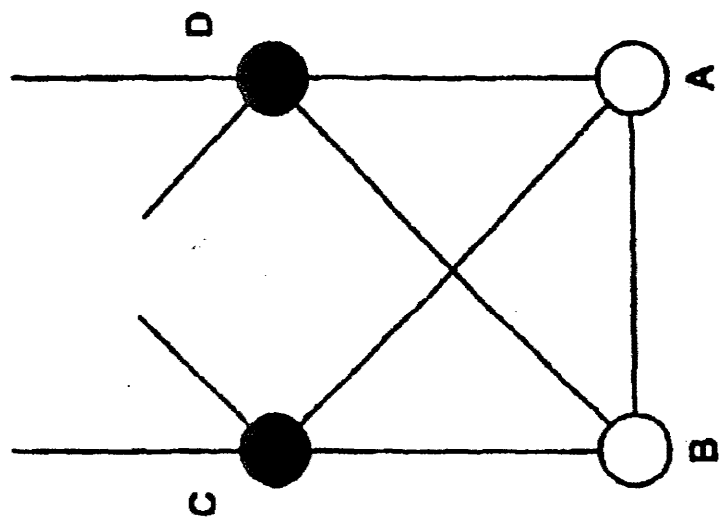


Fig. 5F

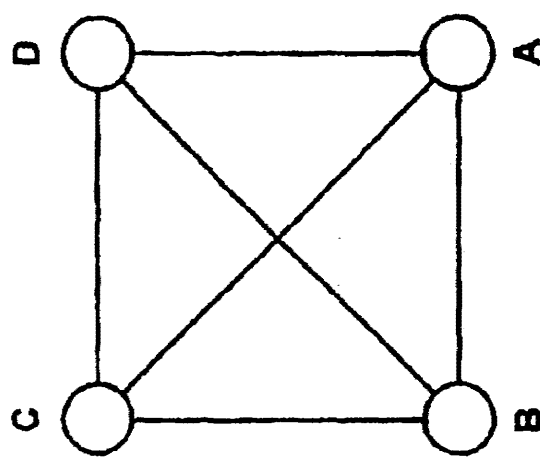


Fig. 5E

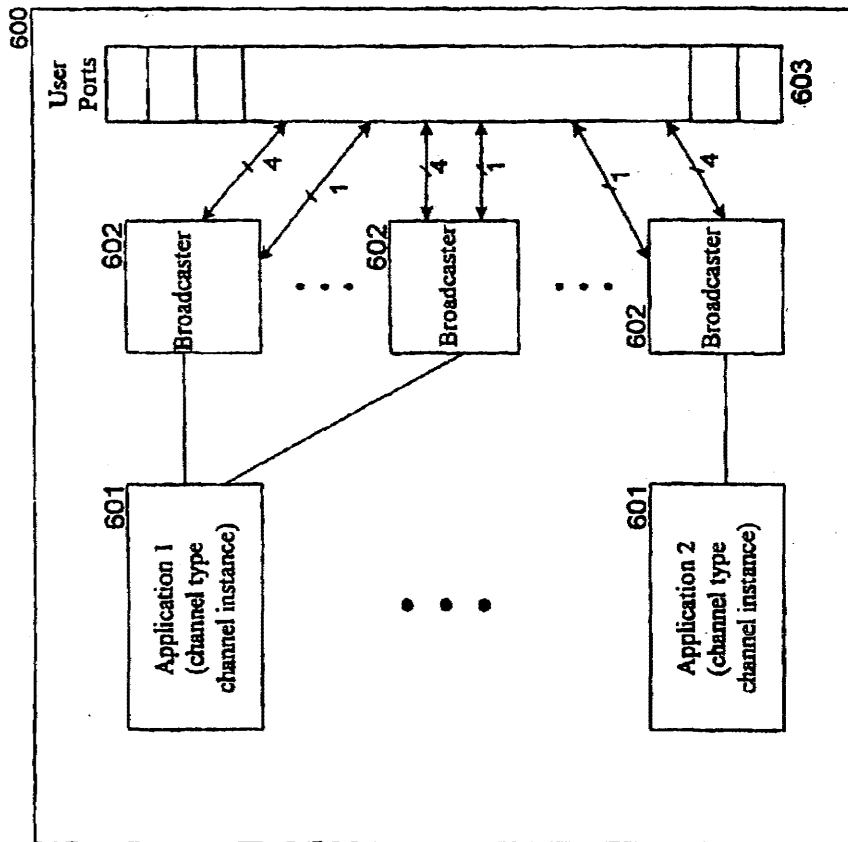


Fig. 6

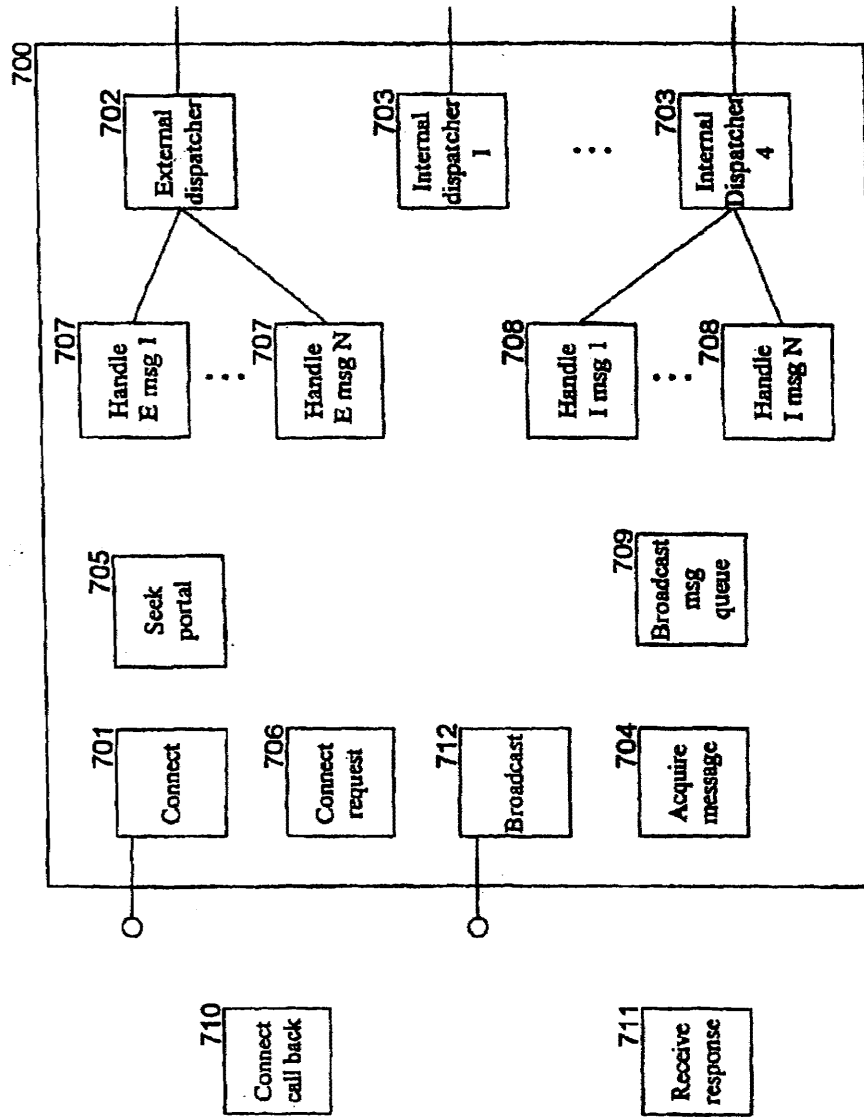
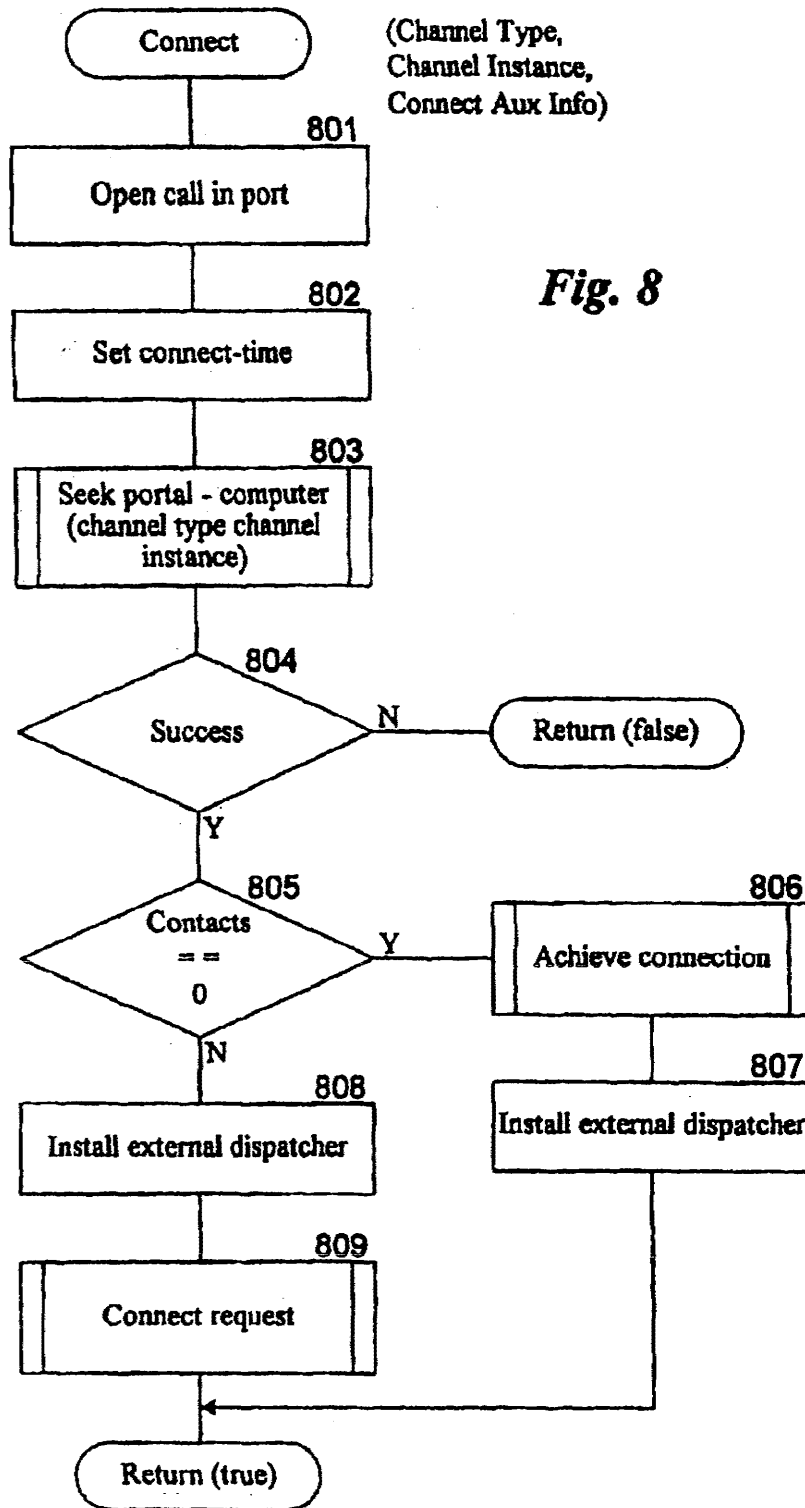
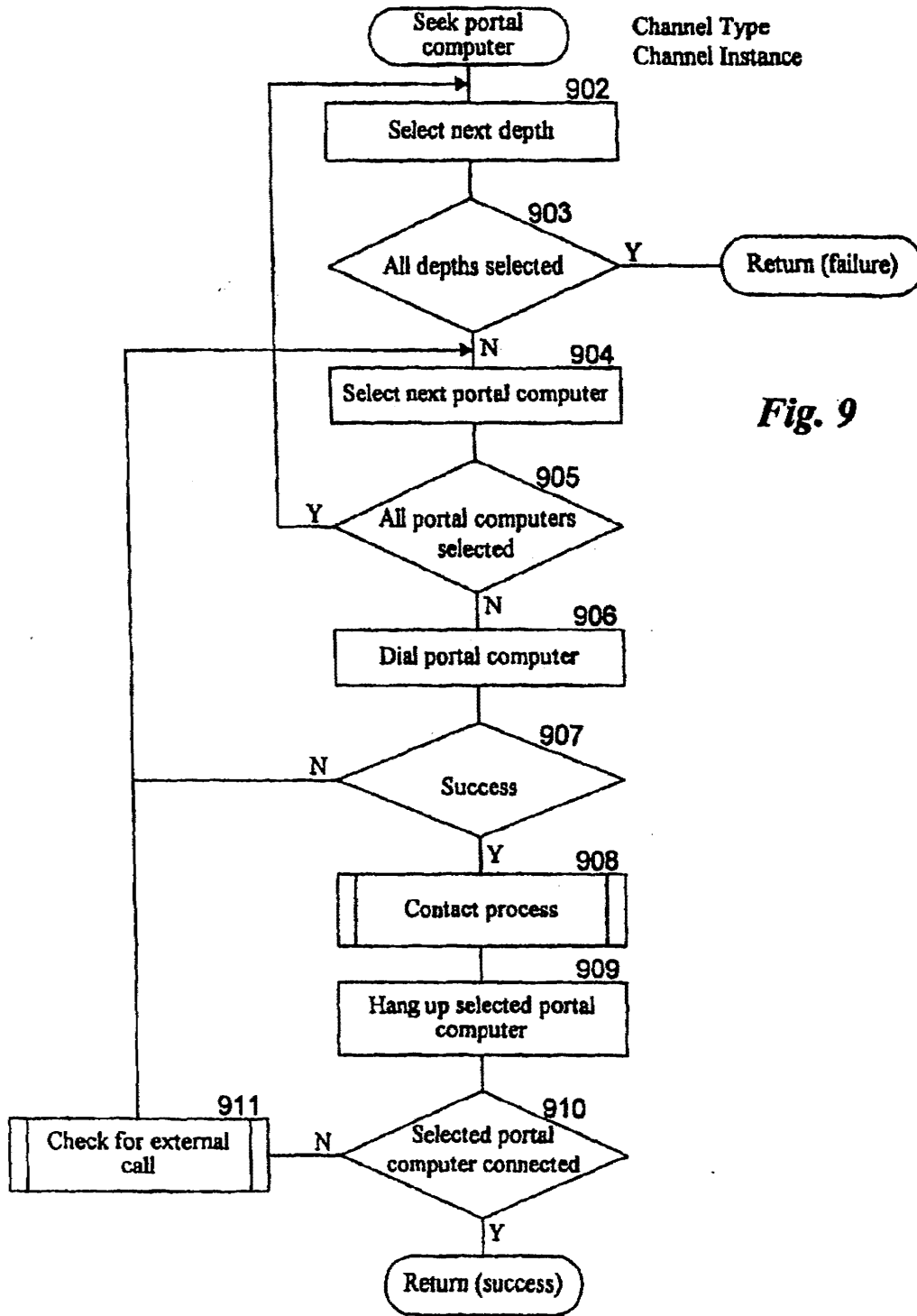


Fig. 7





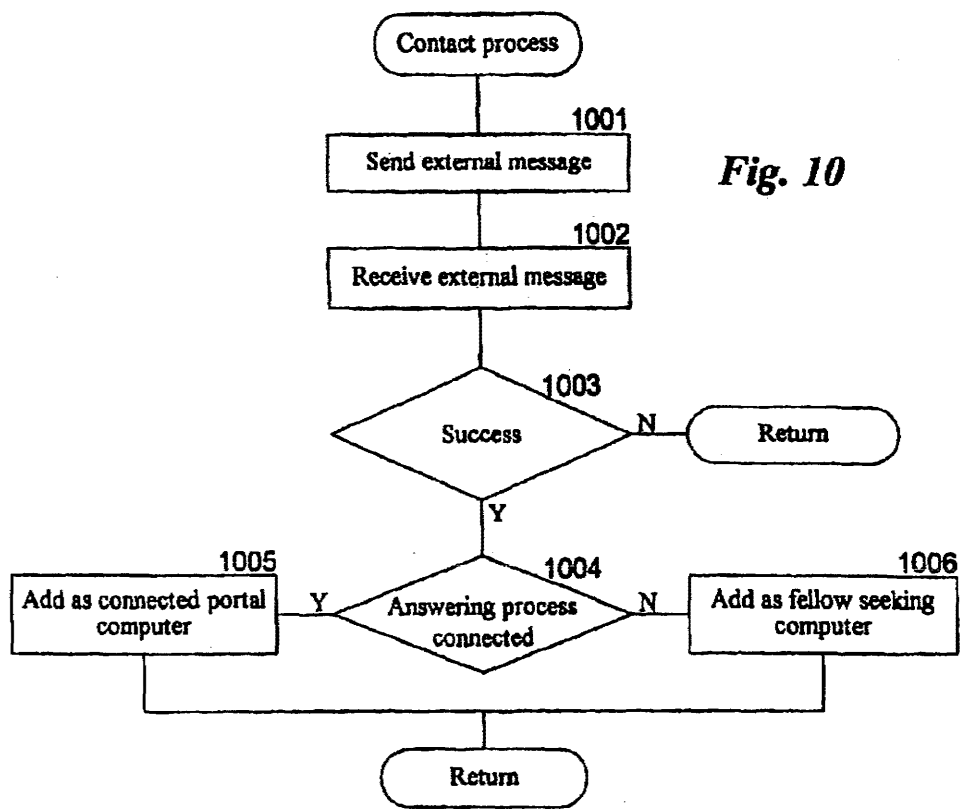


Fig. 11

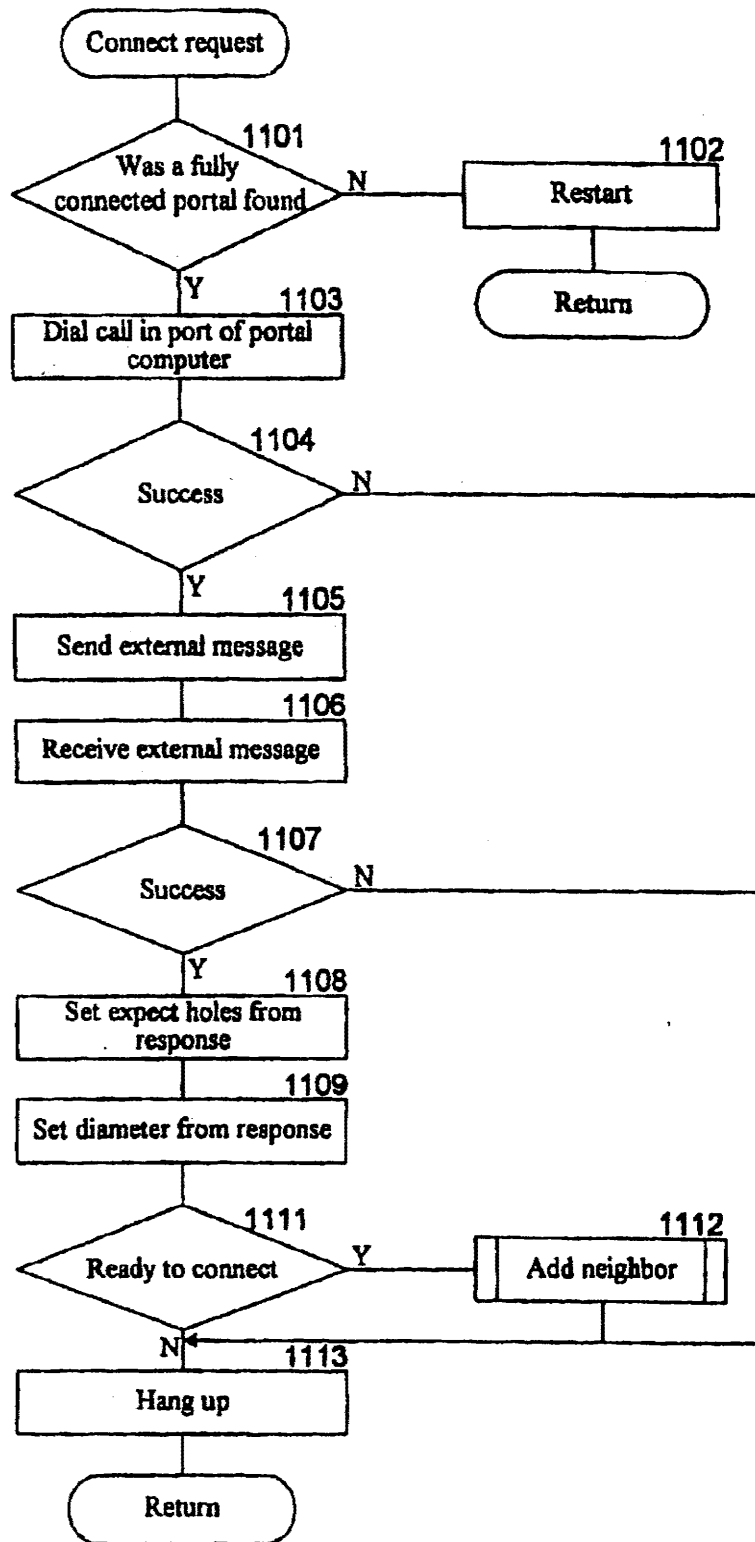
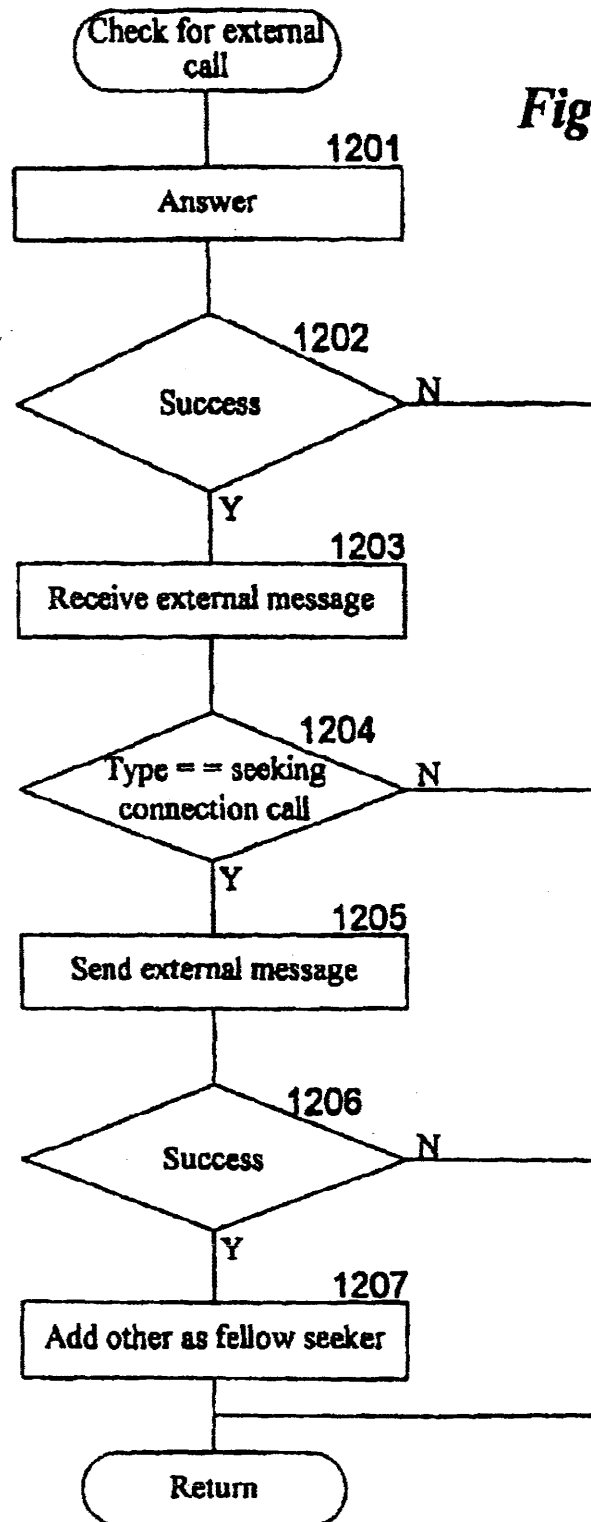


Fig. 12



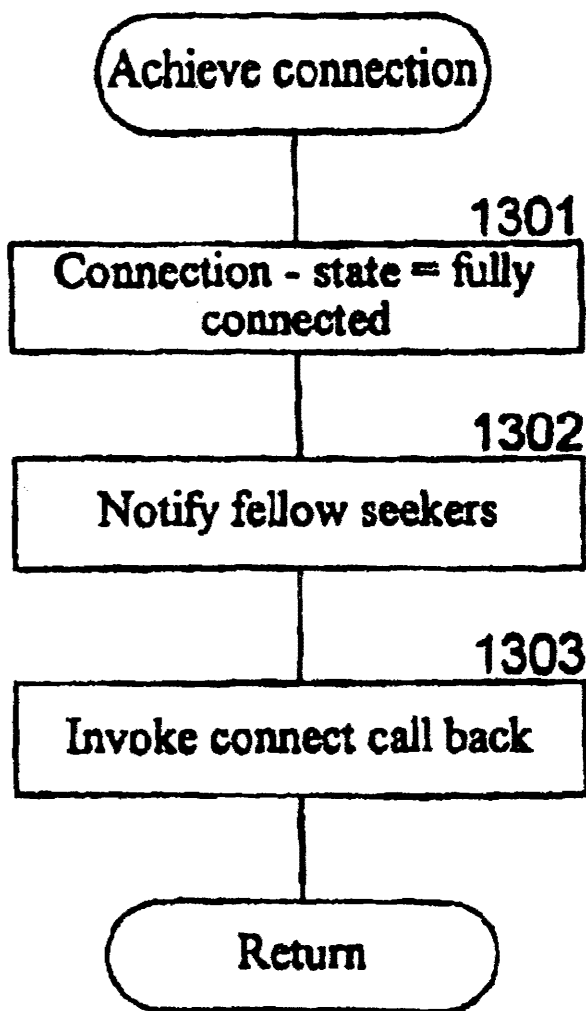


Fig. 13

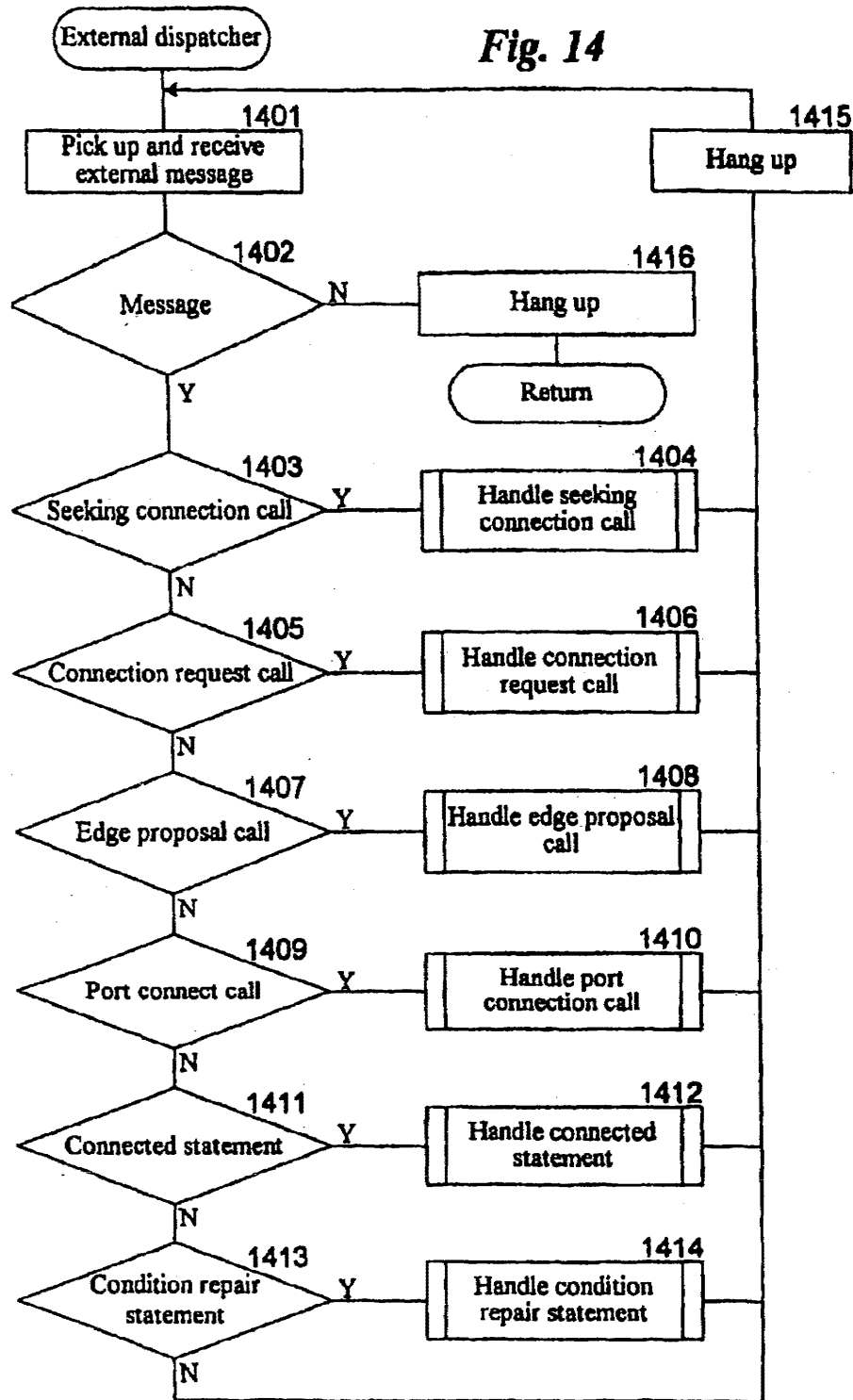
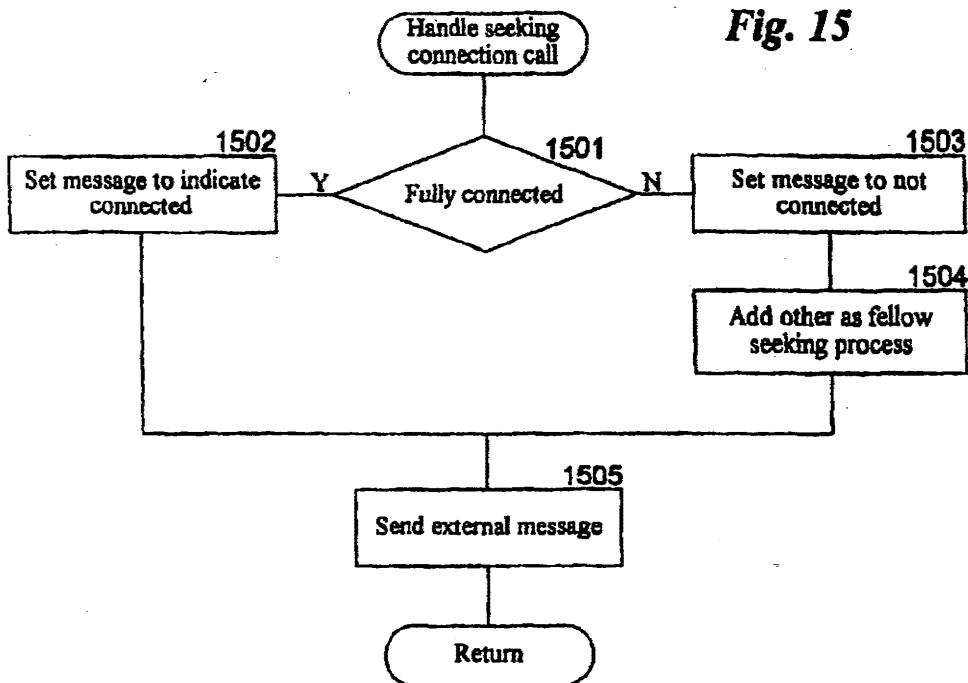


Fig. 15



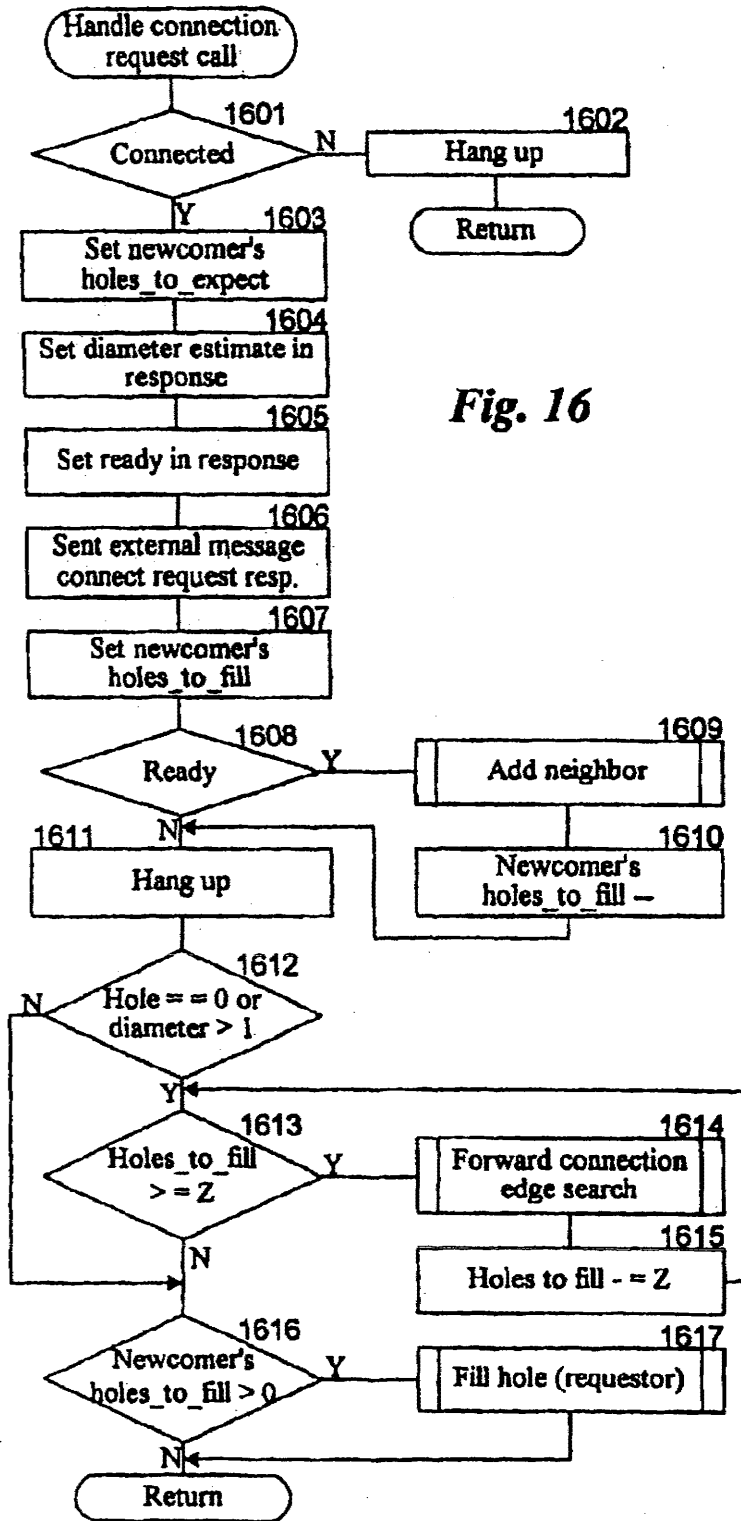


Fig. 16

Fig. 17

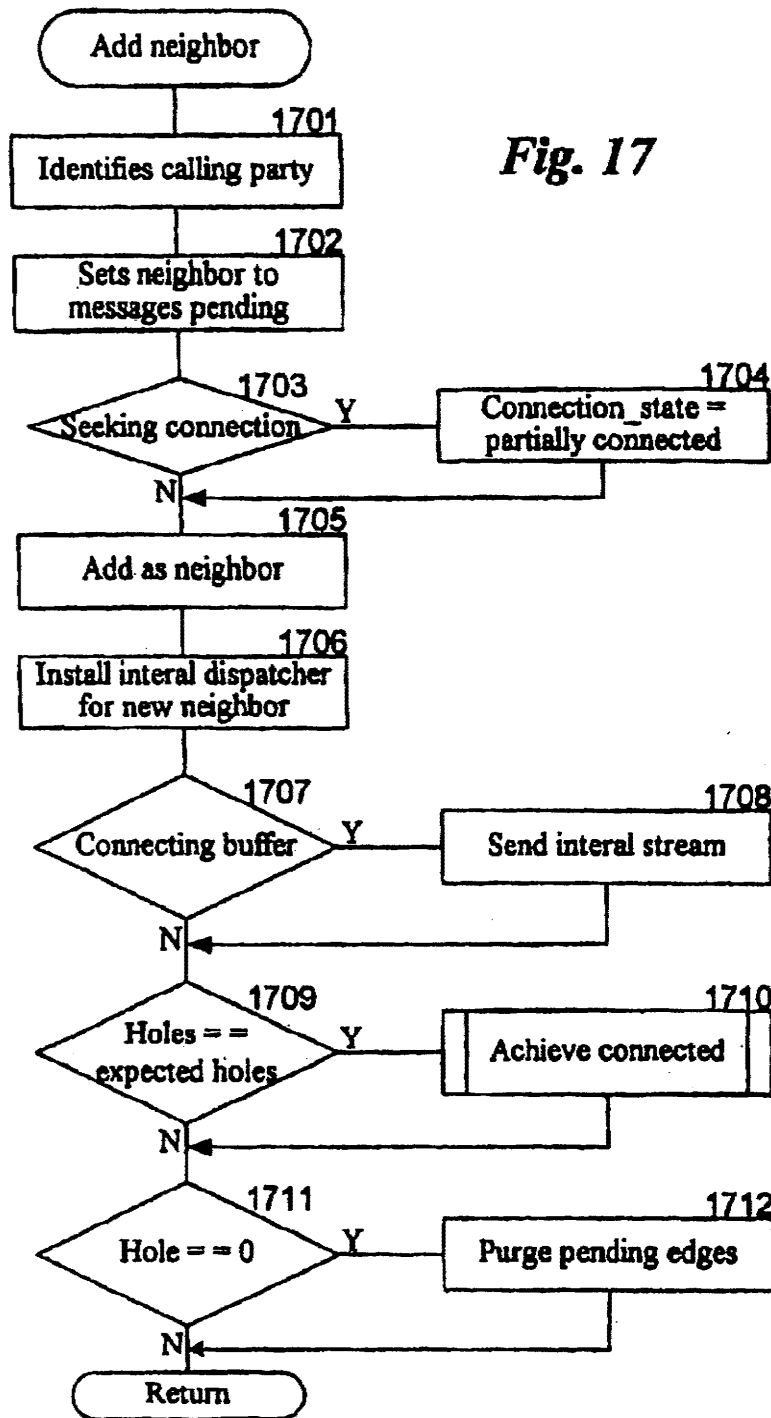
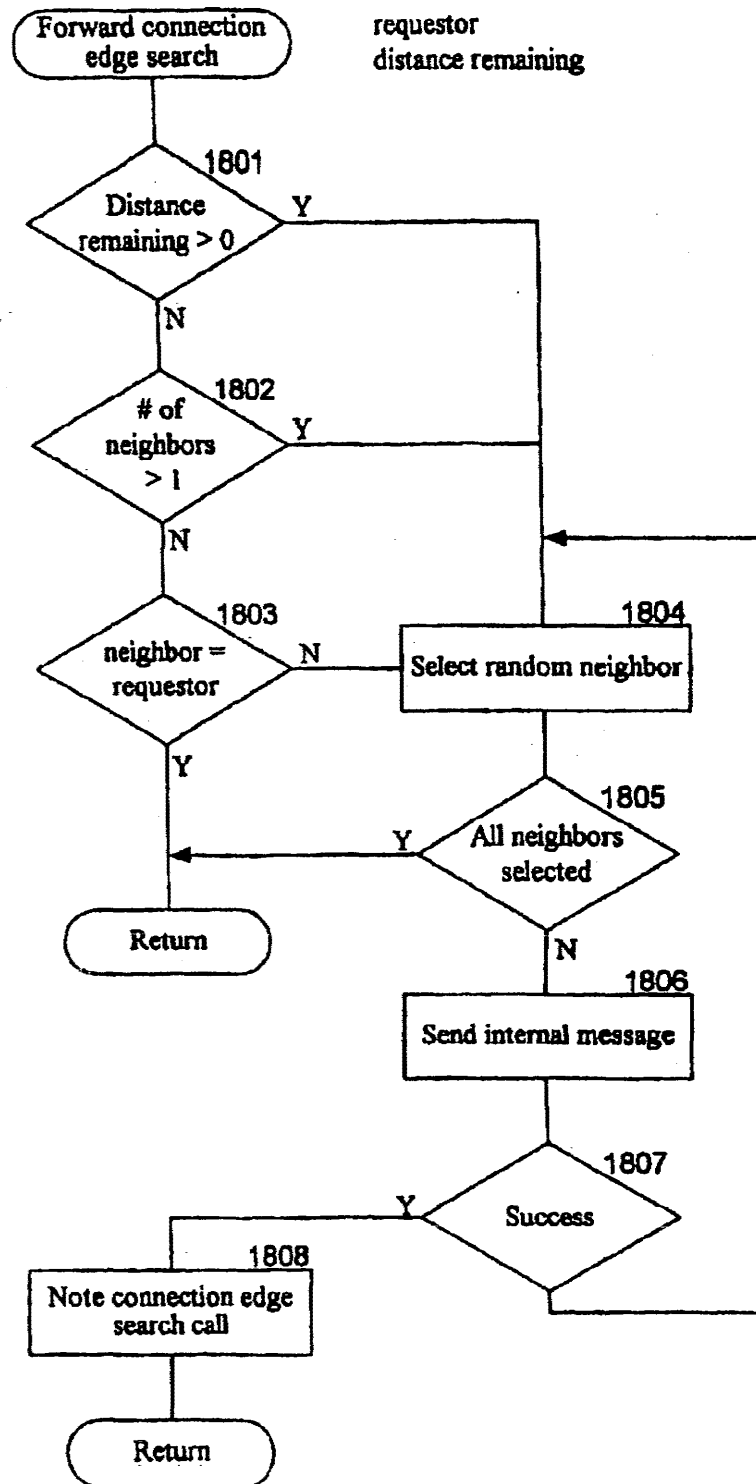
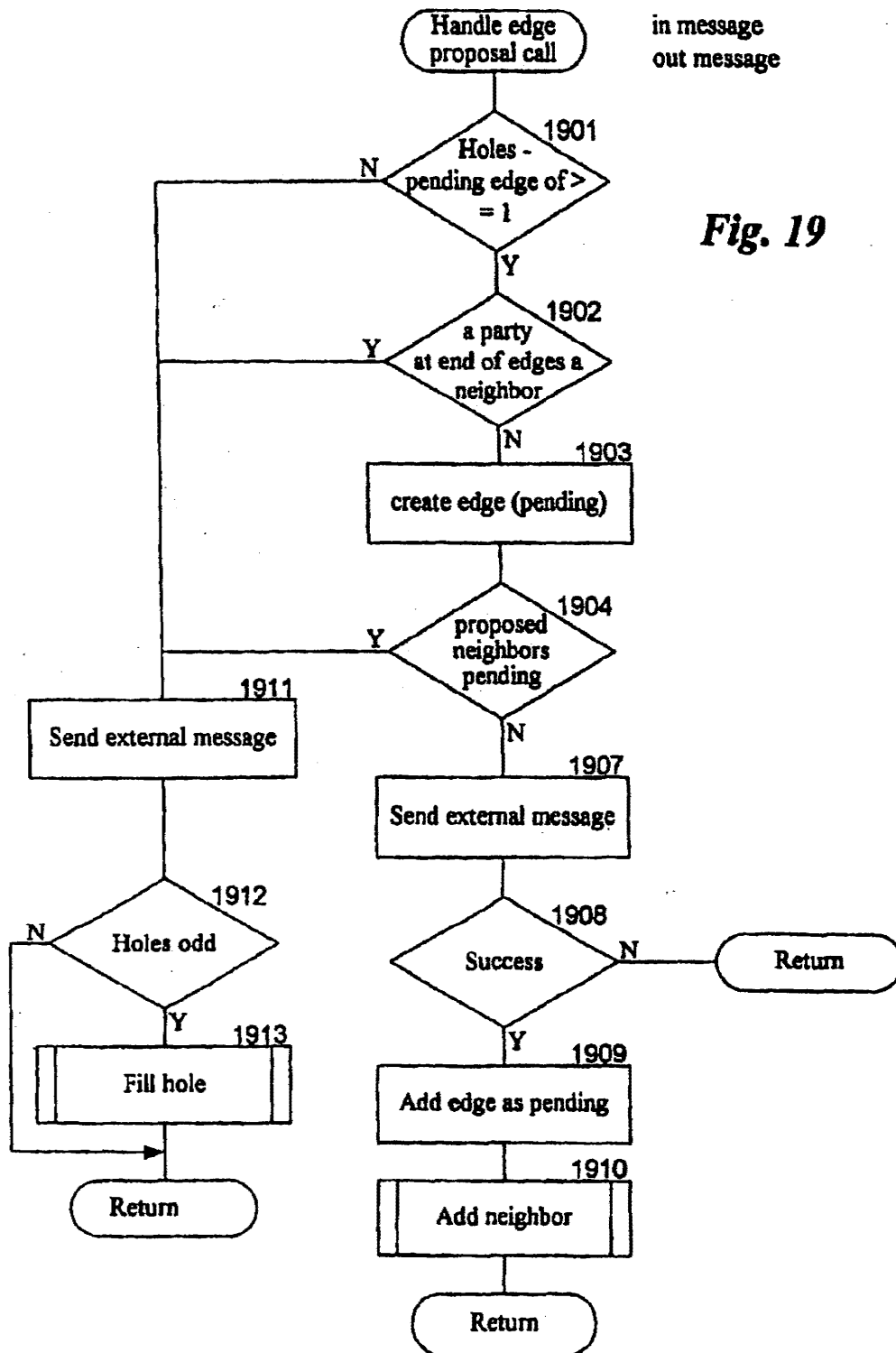


Fig. 18





in message
out message

Fig. 19

Fig. 20

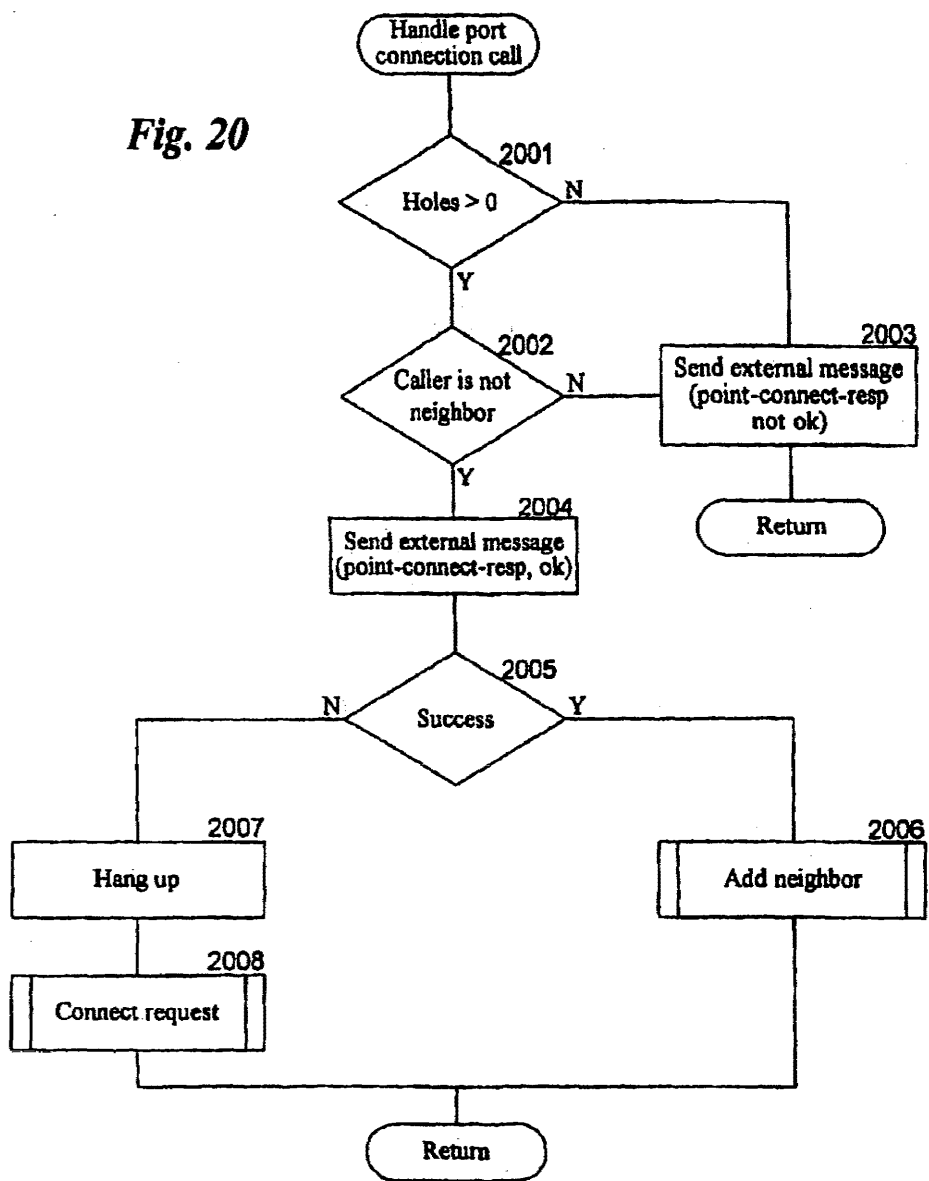


Fig. 21

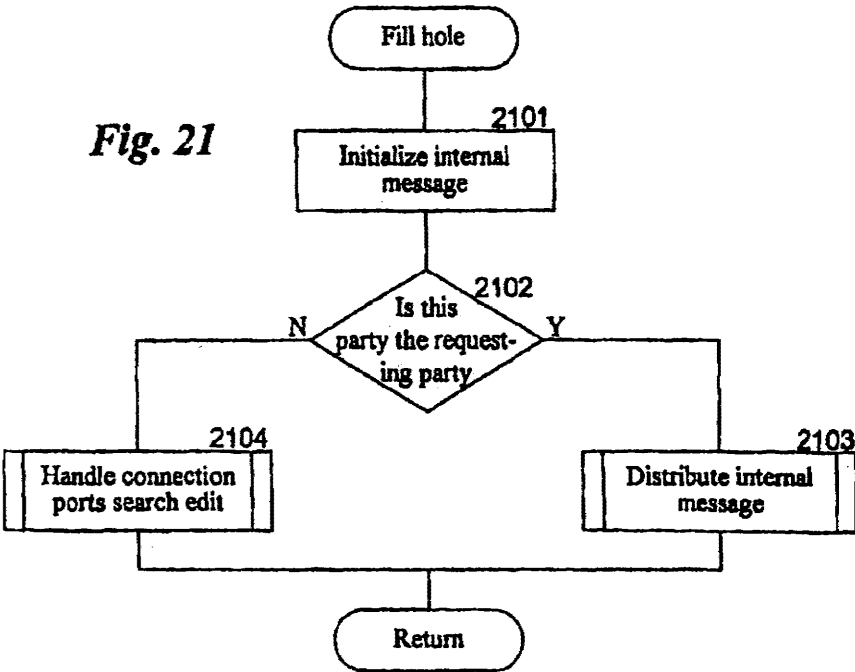


Fig. 22

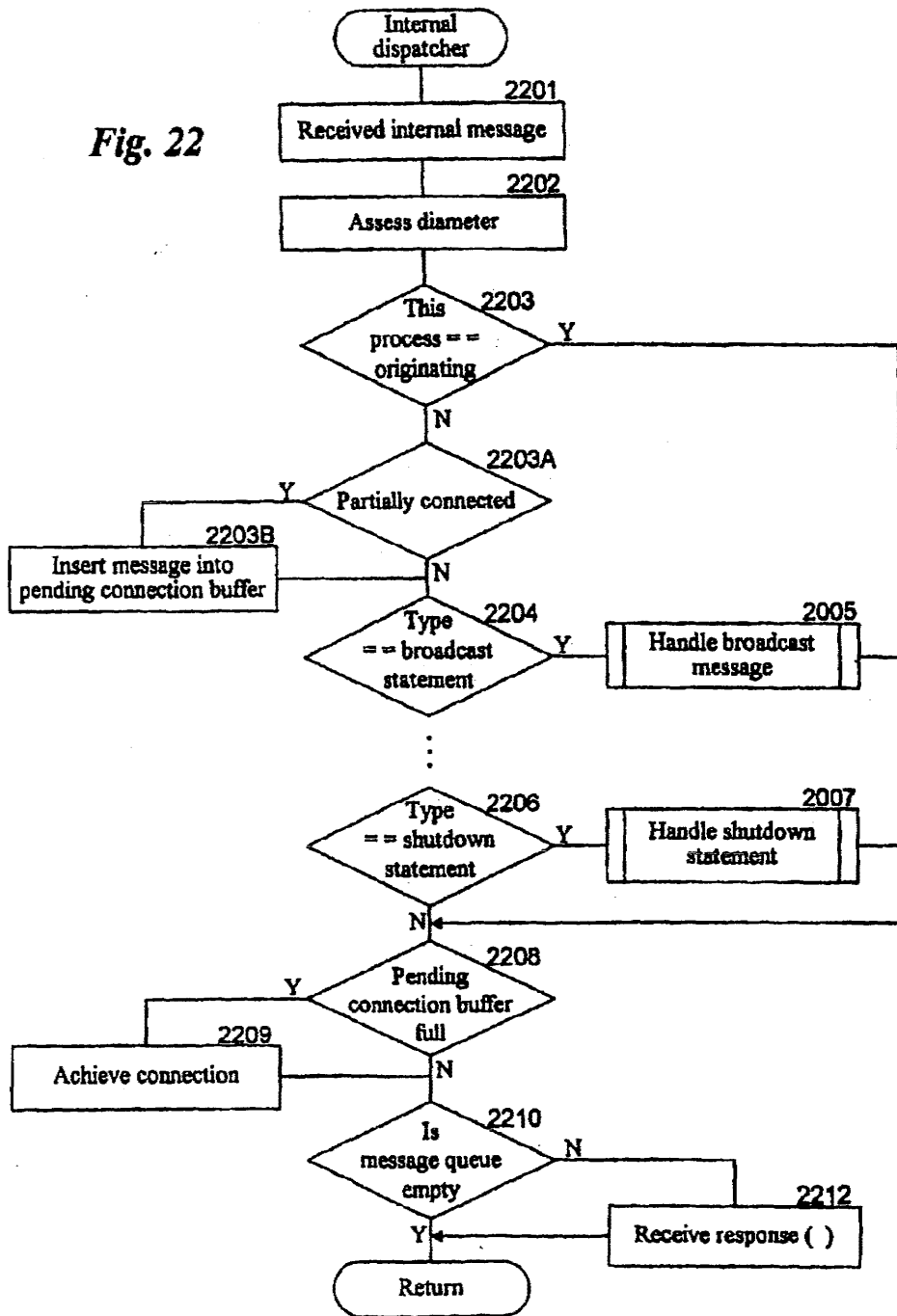


Fig. 23

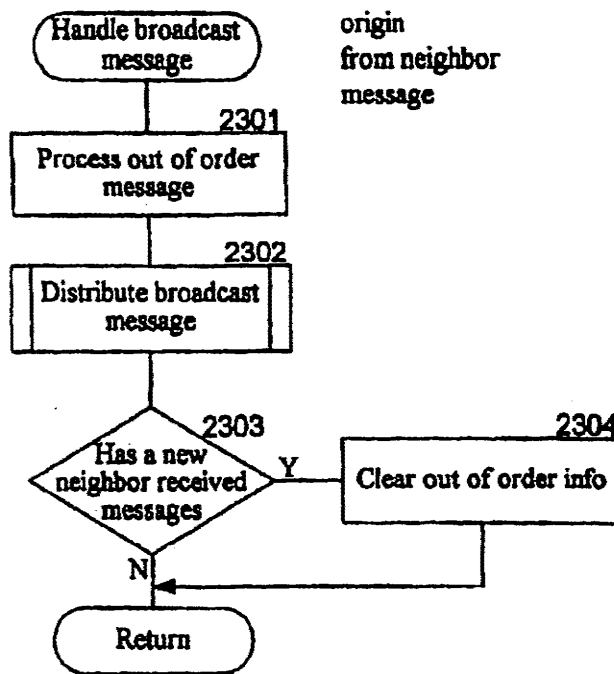
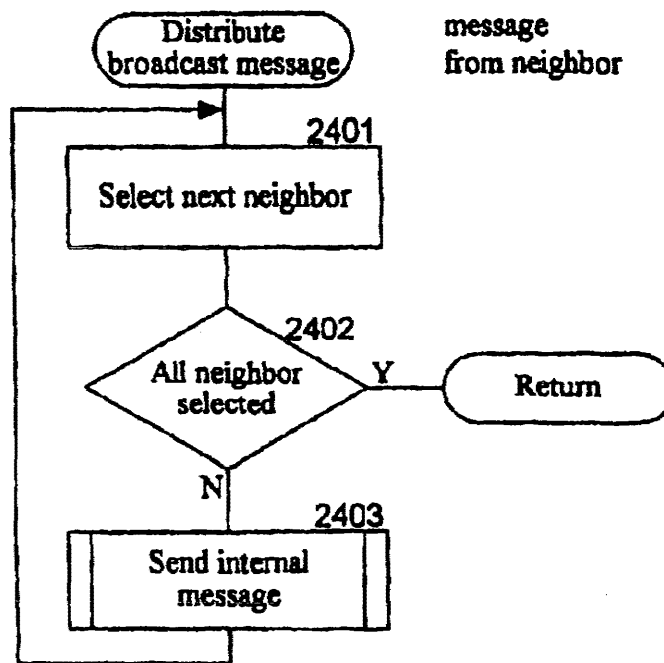


Fig. 24



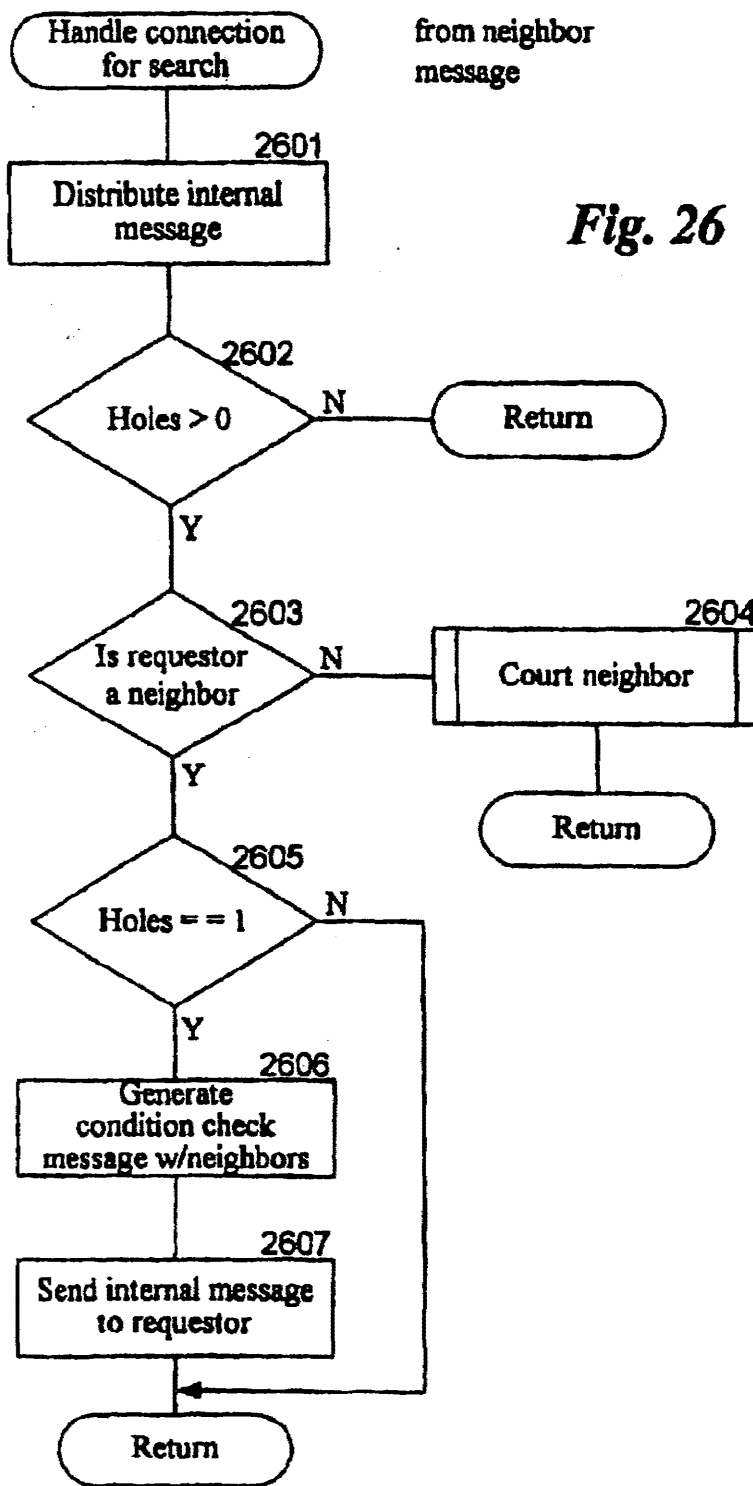


Fig. 26

Fig. 27

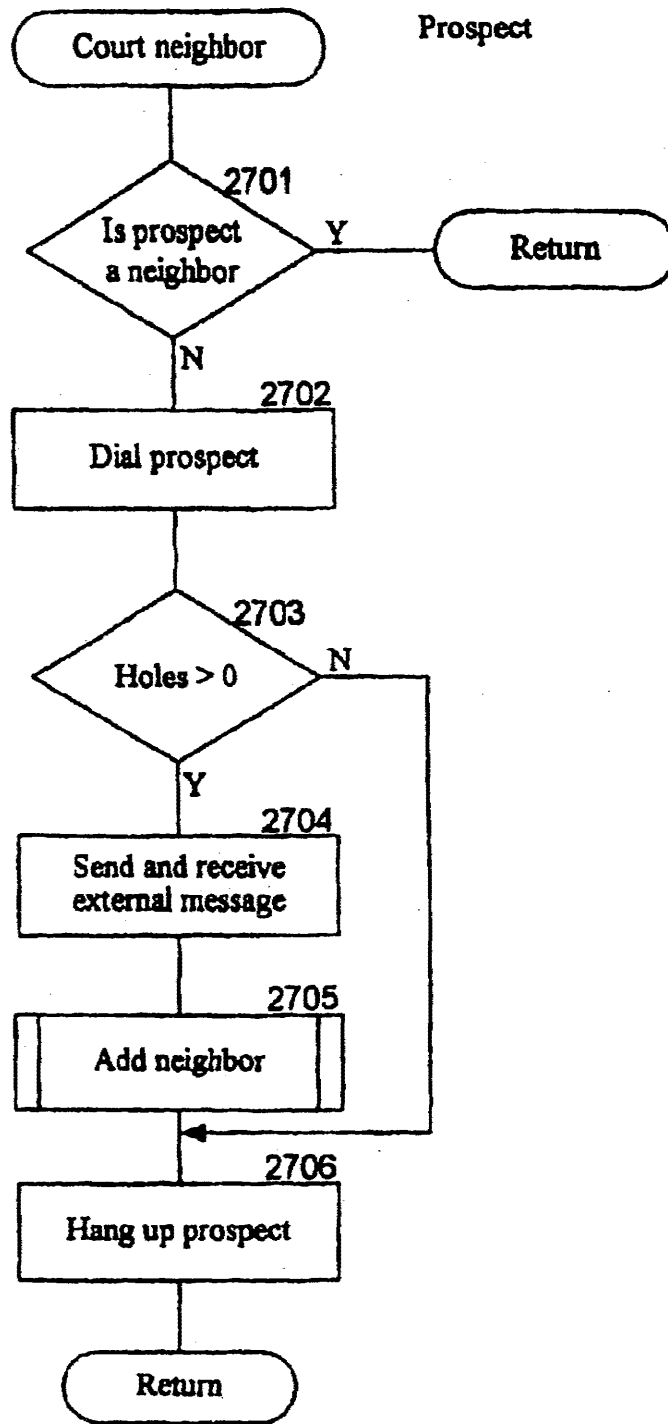
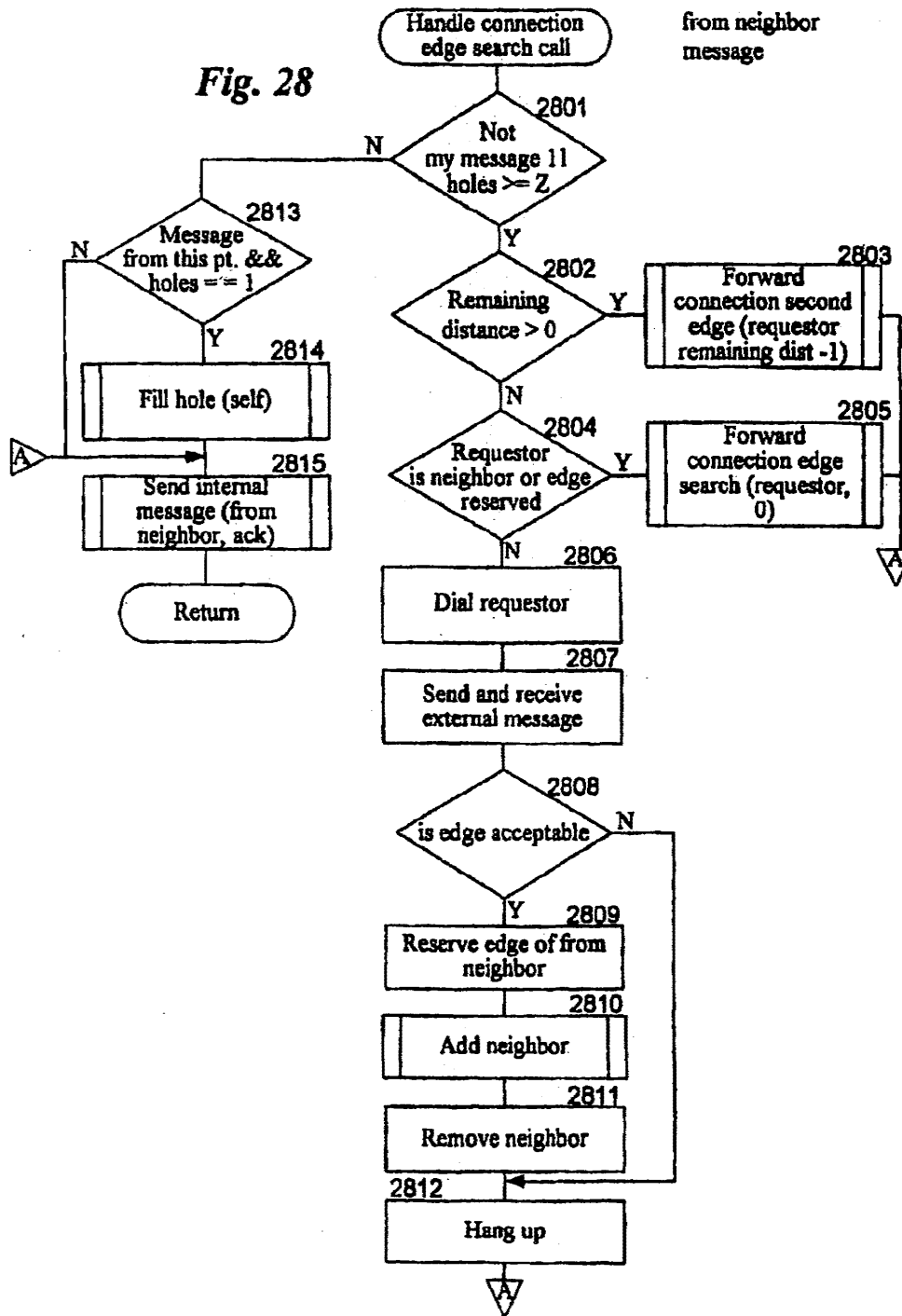


Fig. 28



from neighbor message

Fig. 29

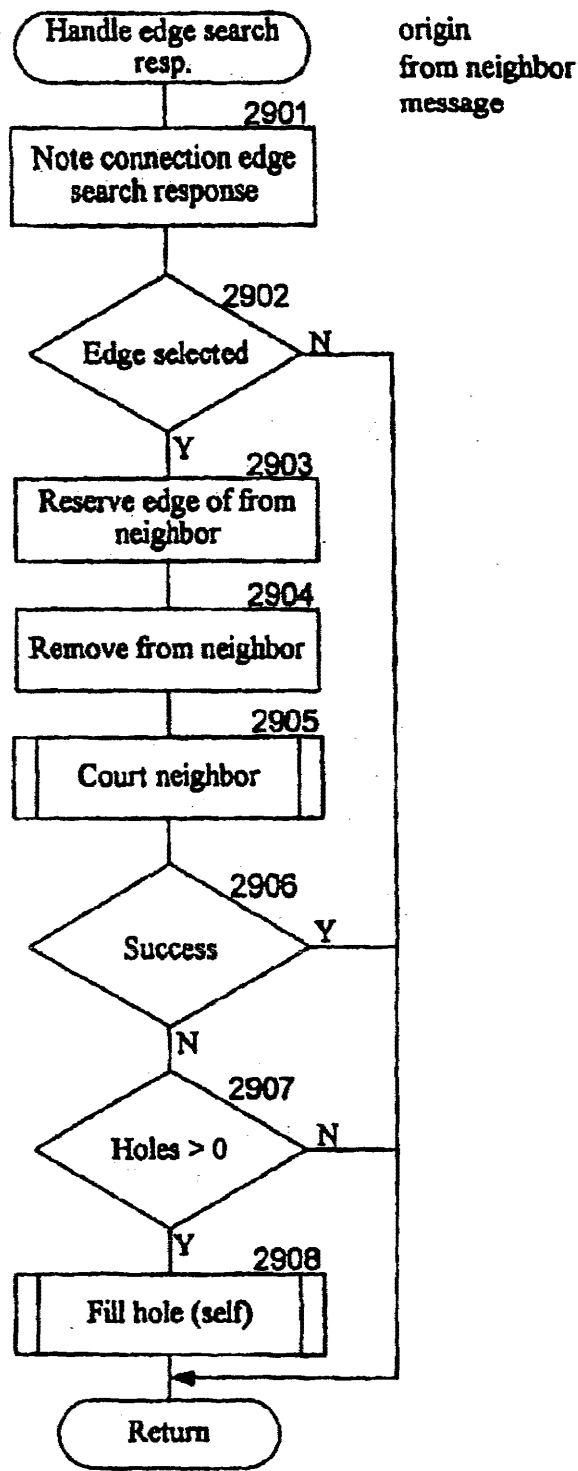


Fig. 30

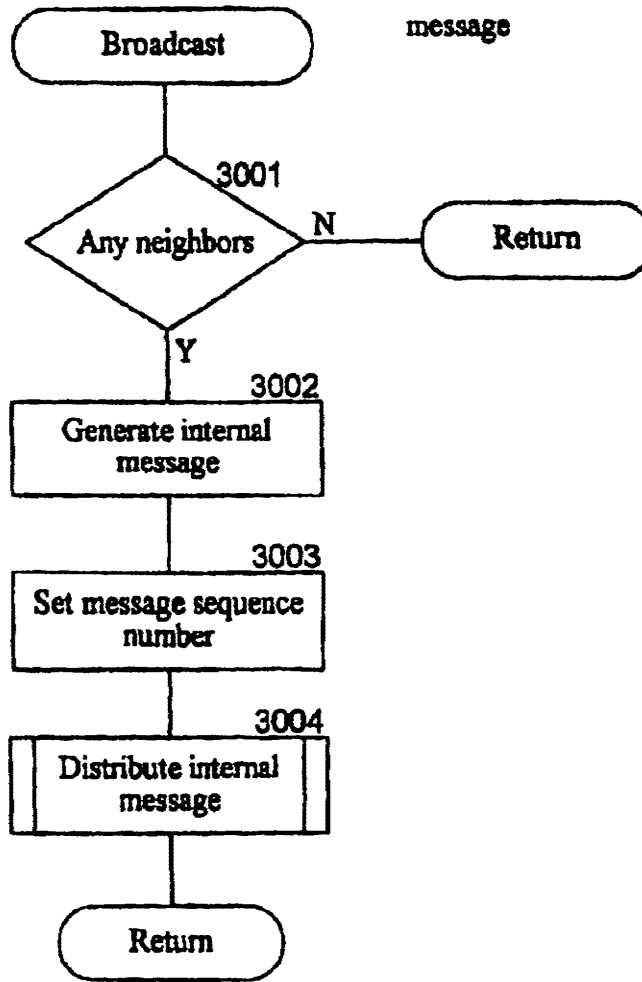


Fig. 31

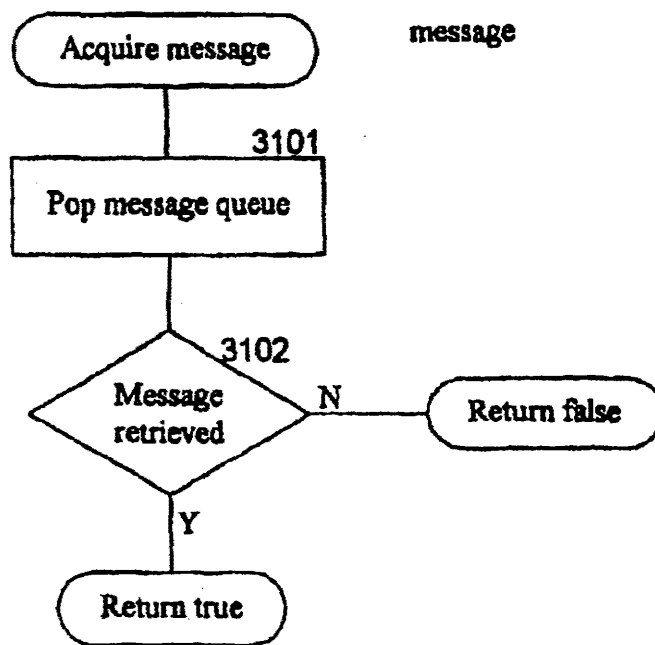


Fig. 32

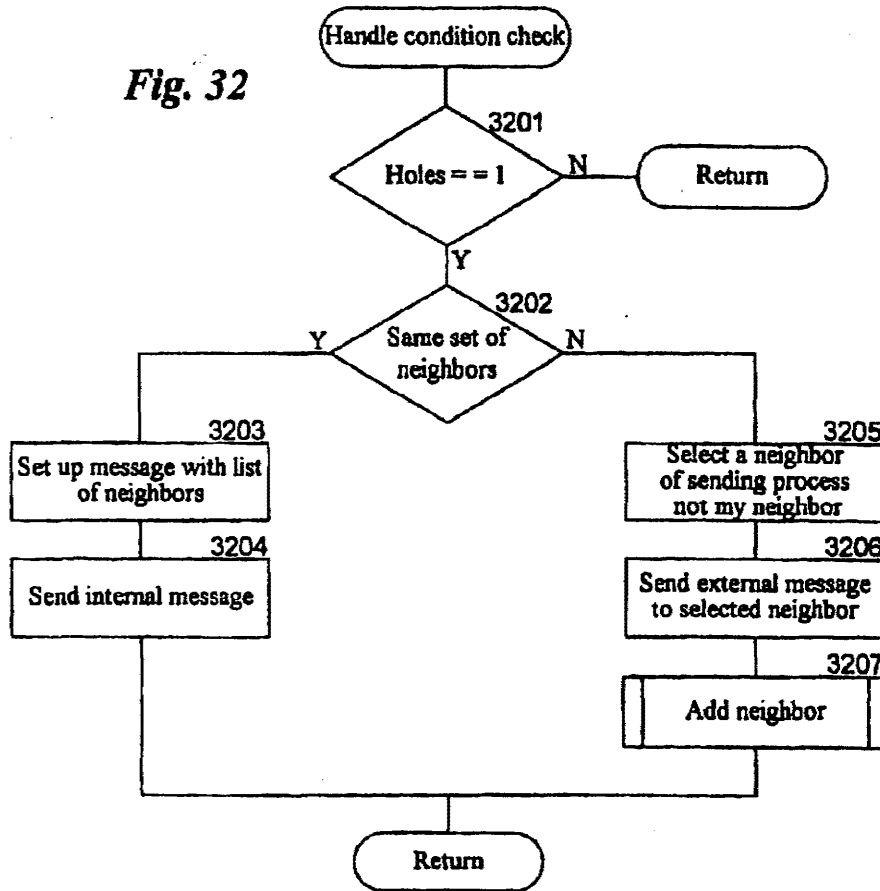
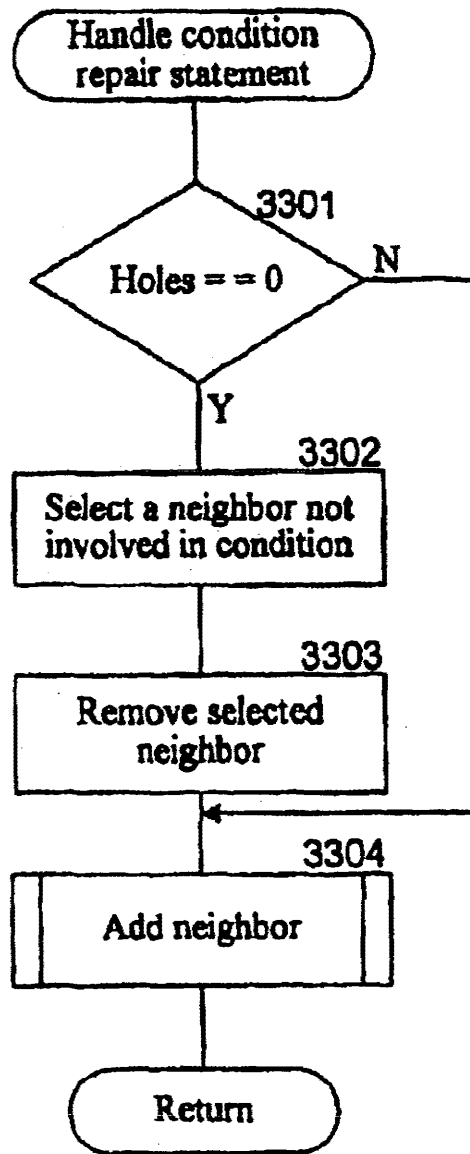
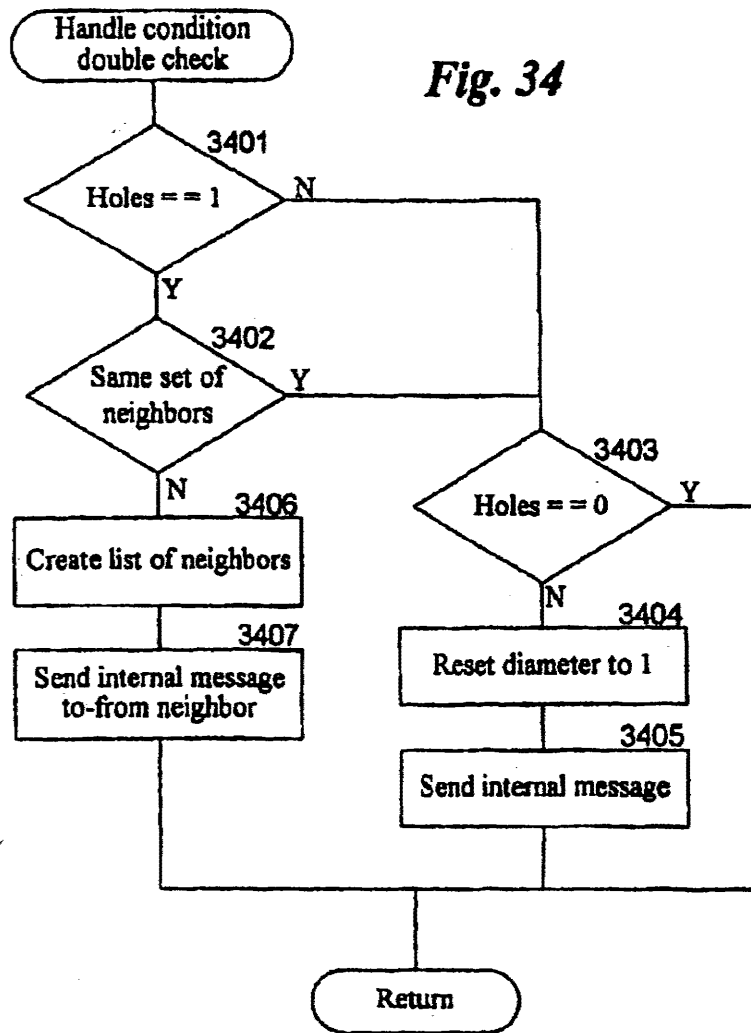


Fig. 33





BROADCASTING NETWORK**CROSS-REFERENCE TO RELATED APPLICATIONS**

This application is related to U.S. patent application Ser. No. 09/629,570, entitled "JOINING A BROADCAST CHANNEL," filed on Jul. 31, 2000 U.S. patent application Ser. No. 09/629,577, "LEAVING A BROADCAST CHANNEL," filed on Jul. 31, 2000 currently patented. U.S. patent application Ser. No. 09/629,575, entitled "BROADCASTING ON A BROADCAST CHANNEL," filed on Jul. 31, 2000; U.S. patent application Ser. No. 09/629,572, entitled "CONTACTING A BROADCAST CHANNEL," filed on Jul. 31, 2000; U.S. patent application Ser. No. 09/629,023, entitled "DISTRIBUTED AUCTION SYSTEM," filed on Jul. 31, 2000 now under appeal. U.S. patent application Ser. No. 09/629,043, entitled "AN INFORMATION DELIVERY SERVICE," filed on Jul. 31, 2000 currently patented; U.S. patent application Ser. No. 09/629,024, entitled "DISTRIBUTED CONFERENCING SYSTEM," filed on Jul. 31, 2000; and U.S. patent application Ser. No. 09/629,042, entitled "DISTRIBUTED GAME ENVIRONMENT," filed on Jul. 31, 2000 currently patented, the disclosures of which are incorporated herein by reference.

TECHNICAL FIELD

The described technology relates generally to a computer network and more particularly, to a broadcast channel for a subset of a computers of an underlying network.

BACKGROUND

There are a wide variety of computer network communications techniques such as point-to-point network protocols, client/server middleware, multicasting network protocols, and peer-to-peer middleware. Each of these communications techniques have their advantages and disadvantages, but none is particularly well suited to the simultaneous sharing of information among computers that are widely distributed. For example, collaborative processing applications, such as a network meeting programs, have a need to distribute information in a timely manner to all participants who may be geographically distributed.

The point-to-point network protocols, such as UNIX pipes, TCP/IP, and UDP, allow processes on different computers to communicate via point-to-point connections. The interconnection of all participants using point-to-point connections, while theoretically possible, does not scale well as a number of participants grows. For example, each participating process would need to manage its direct connections to all other participating processes. Programmers, however, find it very difficult to manage single connections, and management of multiple connections is much more complex. In addition, participating processes may be limited to the number of direct connections that they can support. This limits the number of possible participants in the sharing of information.

The client/server middleware systems provide a server that coordinates the communications between the various clients who are sharing the information. The server functions as a central authority for controlling access to shared resources. Examples of client/server middleware systems include remote procedure calls ("RPC"), database servers, and the common object request broker architecture ("CORBA"). Client/server middleware systems are not par-

ticularly well suited to sharing of information among many participants. In particular, when a client stores information to be shared at the server, each other client would need to poll the server to determine that new information is being shared. Such polling places a very high overhead on the communications network. Alternatively, each client may register a callback with the server, which the server then invokes when new information is available to be shared. Such a callback technique presents a performance bottleneck because a single server needs to call back to each client whenever new information is to be shared. In addition, the reliability of the entire sharing of information depends upon the reliability of the single server. Thus, a failure at a single computer (i.e., the server) would prevent communications between any of the clients.

The multicasting network protocols allow the sending of broadcast messages to multiple recipients of a network. The current implementations of such multicasting network protocols tend to place an unacceptable overhead on the underlying network. For example, UDP multicasting would swamp the Internet when trying to locate all possible participants. IP multicasting has other problems that include needing special-purpose infrastructure (e.g., routers) to support the sharing of information efficiently.

The peer-to-peer middleware communications systems rely on a multicasting network protocol or a graph of point-to-point network protocols. Such peer-to-peer middleware is provided by the T.120 Internet standard, which is used in such products as Data Connection's D.C.-share and Microsoft's NetMeeting. These peer-to-peer middleware systems rely upon a user to assemble a point-to-point graph of the connections used for sharing the information. Thus, it is neither suitable nor desirable to use peer-to-peer middleware systems when more than a small number of participants is desired. In addition, the underlying architecture of the T.120 Internet standard is a tree structure, which relies on the root node of the tree for reliability of the entire network. That is, each message must pass through the root node in order to be received by all participants.

It would be desirable to have a reliable communications network that is suitable for the simultaneous sharing of information among a large number of the processes that are widely distributed.

SUMMARY OF THE INVENTION

Embodiments of the invention deal with a non-routing table based method for broadcasting messages in a network. More specifically, a network in which each participant has at least three neighbor participants broadcasts data through each of its connections to neighbor participants, which in turn send the data that it receives to its other neighbor participants. The data is numbered sequentially so that data that is received out of order can be queued and rearranged.

Communication within the broadcast channel is controlled by a contact module and by a join module. The contact module locates a portal computer and requests the located portal computer to provide an indication of neighbor participants to which the participant can be connected. The join module receives the indication of the neighbor participants and establishes a connection between the seeking participant and each of the indicated neighbor participants.

Each participant in the network is connected to neighbor participants, and the participants and connections between them form an m-regular graph, where m is greater than 2. In addition, when a participant receives data from a neighbor participant, it sends the data to its other neighbor participants.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a graph that is 4-regular and 4-connected which represents a broadcast channel.

FIG. 2 illustrates a graph representing 20 computers connected to a broadcast channel.

FIGS. 3A and 3B illustrate the process of connecting a new computer Z to the broadcast channel.

FIG. 4A illustrates the broadcast channel of FIG. 1 with an added computer.

FIG. 4B illustrates the broadcast channel of FIG. 4A with an added computer.

FIG. 4C also illustrates the broadcast channel of FIG. 4A with an added computer.

FIG. 5A illustrates the disconnecting of a computer from the broadcast channel in a planned manner.

FIG. 5B illustrates the disconnecting of a computer from the broadcast channel in an unplanned manner.

FIG. 5C illustrates the neighbors with empty ports condition.

FIG. 5D illustrates two computers that are not neighbors who now have empty ports.

FIG. 5E illustrates the neighbors with empty ports condition in the small regime.

FIG. 5F illustrates the situation of FIG. 5E when in the large regime.

FIG. 6 is a block diagram illustrating components of a computer that is connected to a broadcast channel.

FIG. 7 is a block diagram illustrating the sub-components of the broadcaster component in one embodiment.

FIG. 8 is a flow diagram illustrating the processing of the connect routine in one embodiment.

FIG. 9 is a flow diagram illustrating the processing of the seek portal computer routine in one embodiment.

FIG. 10 is a flow diagram illustrating the processing of the contact process routine in one embodiment.

FIG. 11 is a flow diagram illustrating the processing of the connect request routine in one embodiment.

FIG. 12 is a flow diagram of the processing of the check for external call routine in one embodiment.

FIG. 13 is a flow diagram of the processing of the achieve connection routine in one embodiment.

FIG. 14 is a flow diagram illustrating the processing of the external dispatcher routine in one embodiment.

FIG. 15 is a flow diagram illustrating the processing of the handle seeking connection call routine in one embodiment.

FIG. 16 is a flow diagram illustrating processing of the handle connection request call routine in one embodiment.

FIG. 17 is a flow diagram illustrating the processing of the add neighbor routine in one embodiment.

FIG. 18 is a flow diagram illustrating the processing of the forward connection edge search routine in one embodiment.

FIG. 19 is a flow diagram illustrating the processing of the handle edge proposal call routine.

FIG. 20 is a flow diagram illustrating the processing of the handle port connection call routine in one embodiment.

FIG. 21 is a flow diagram illustrating the processing of the fill hole routine in one embodiment.

FIG. 22 is a flow diagram illustrating the processing of the internal dispatcher routine in one embodiment.

FIG. 23 is a flow diagram illustrating the processing of the handle broadcast message routine in one embodiment.

FIG. 24 is a flow diagram illustrating the processing of the distribute broadcast message routine in one embodiment.

FIG. 26 is a flow diagram illustrating the processing of the handle connection port search statement routine in one embodiment.

FIG. 27 is a flow diagram illustrating the processing of the court neighbor routine in one embodiment.

FIG. 28 is a flow diagram illustrating the processing of the handle connection edge search call routine in one embodiment.

FIG. 29 is a flow diagram illustrating the processing of the handle connection edge search response routine in one embodiment.

FIG. 30 is a flow diagram illustrating the processing of the broadcast routine in one embodiment.

FIG. 31 is a flow diagram illustrating the processing of the acquire message routine in one embodiment.

FIG. 32 is a flow diagram illustrating processing of the handle condition check message in one embodiment.

FIG. 33 is a flow diagram illustrating processing of the handle condition repair statement routine in one embodiment.

FIG. 34 is a flow diagram illustrating the processing of the handle condition double check routine.

DETAILED DESCRIPTION

A broadcast technique in which a broadcast channel overlays a point-to-point communications network is provided. The broadcasting of a message over the broadcast channel is effectively a multicast to those computers of the network that are currently connected to the broadcast channel. In one embodiment, the broadcast technique provides a logical broadcast channel to which host computers through their executing processes can be connected. Each computer that is connected to the broadcast channel can broadcast messages onto and receive messages off of the broadcast channel. Each computer that is connected to the broadcast channel receives all messages that are broadcast while it is connected. The logical broadcast channel is implemented using an underlying network system (e.g., the Internet) that allows each computer connected to the underlying network system to send messages to each other connected computer using each computer's address. Thus, the broadcast technique effectively provides a broadcast channel using an underlying network system that sends messages on a point-to-point basis.

The broadcast technique overlays the underlying network system with a graph of point-to-point connections (i.e., edges) between host computers (i.e., nodes) through which the broadcast channel is implemented. In one embodiment, each computer is connected to four other computers, referred to as neighbors. (Actually, a process executing on a computer is connected to four other processes executing on this or four other computers.) To broadcast a message, the originating computer sends the message to each of its neighbors using its point-to-point connections. Each computer that receives the message then sends the message to its three other neighbors using the point-to-point connections. In this way, the message is propagated to each computer using the underlying network to effect the broadcasting of the message to each computer over a logical broadcast channel. A graph in which each node is connected to four other nodes is referred to as a 4-regular graph. The use of a 4-regular graph means that a computer would become disconnected from the broadcast channel only if all four of

the connections to its neighbors fail. The graph used by the broadcast technique also has the property that it would take a failure of four computers to divide the graph into disjoint sub-graphs, that is two separate broadcast channels. This property is referred to as being 4-connected. Thus, the graph is both 4-regular and 4-connected.

FIG. 1 illustrates a graph that is 4-regular and 4-connected which represents the broadcast channel. Each of the nine nodes A-I represents a computer that is connected to the broadcast channel, and each of the edges represents an "edge" connection between two computers of the broadcast channel. The time it takes to broadcast a message to each computer on the broadcast channel depends on the speed of the connections between the computers and the number of connections between the originating computer and each other computer on the broadcast channel. The minimum number of connections that a message would need to traverse between each pair of computers is the "distance" between the computers (i.e., the shortest path between the two nodes of the graph). For example, the distance between computers A and F is one because computer A is directly connected to computer F. The distance between computers A and B is two because there is no direct connection between computers A and B, but computer F is directly connected to computer B. Thus, a message originating at computer A would be sent directly to computer F, and then sent from computer F to computer B. The maximum of the distances between the computers is the "diameter" of broadcast channel. The diameter of the broadcast channel represented by FIG. 1 is two. That is, a message sent by any computer would traverse no more than two connections to reach every other computer. FIG. 2 illustrates a graph representing 20 computers connected to a broadcast channel. The diameter of this broadcast channel is 4. In particular, the shortest path between computers 1 and 3 contains four connections (1-12, 12-15, 15-18, and 18-3).

The broadcast technique includes (1) the connecting of computers to the broadcast channel (i.e., composing the graph), (2) the broadcasting of messages over the broadcast channel (i.e., broadcasting through the graph), and (3) the disconnecting of computers from the broadcast channel (i.e., decomposing the graph) composing the graph.

Composing the Graph

To connect to the broadcast channel, the computer seeking the connection first locates a computer that is currently fully connected to the broadcast channel and then establishes a connection with four of the computers that are already connected to the broadcast channel. (This assumes that there are at least four computers already connected to the broadcast channel. When there are fewer than five computers connected, the broadcast channel cannot be a 4-regular graph. In such a case, the broadcast channel is considered to be in a "small regime." The broadcast technique for the small regime is described below in detail. When five or more computers are connected, the broadcast channel is considered to be in the "large regime." This description assumes that the broadcast channel is in the large regime, unless specified otherwise.) Thus, the process of connecting to the broadcast channel includes locating the broadcast channel, identifying the neighbors for the connecting computer, and then connecting to each identified neighbor. Each computer is aware of one or more "portal computers" through which that computer may locate the broadcast channel. A seeking computer locates the broadcast channel by contacting the portal computers until it finds one that is currently fully connected to the broadcast channel. The found portal com-

puter then directs the identifying of four computers (i.e., to be the seeking computer's neighbors) to which the seeking computer is to connect. Each of these four computers then cooperates with the seeking computer to effect the connecting of the seeking computer to the broadcast channel. A computer that has started the process of locating a portal computer, but does not yet have a neighbor, is in the "seeking connection state." A computer that is connected to at least one neighbor, but not yet four neighbors, is in the "partially connected state." A computer that is currently, or has been, previously connected to four neighbors is in the "fully connected state."

Since the broadcast channel is a 4-regular graph, each of the identified computers is already connected to four computers. Thus, some connections between computers need to be broken so that the seeking computer can connect to four computers. In one embodiment, the broadcast technique identifies two pairs of computers that are currently connected to each other. Each of these pairs of computers breaks the connection between them, and then each of the four computers (two from each pair) connects to the seeking computer. FIGS. 3A and 3B illustrate the process of a new computer Z connecting to the broadcast channel. FIG. 3A illustrates the broadcast channel before computer Z is connected. The pairs of computers B and E and computers C and D are the two pairs that are identified as the neighbors for the new computer Z. The connections between each of these pairs is broken, and a connection between computer Z and each of computers B, C, D, and E is established as indicated by FIG. 3B. The process of breaking the connection between two neighbors and reconnecting each of the former neighbors to another computer is referred to as "edge pinning" as the edge between two nodes may be considered to be stretched and pinned to a new node.

Each computer connected to the broadcast channel allocates five communications ports for communicating with other computers. Four of the ports are referred to as "internal" ports because they are the ports through which the messages of the broadcast channels are sent. The connections between internal ports of neighbors are referred to as "internal" connections. Thus, the internal connections of the broadcast channel form the 4-regular and 4-connected graph. The fifth port is referred to as an "external" port because it is used for sending non-broadcast messages between two computers. Neighbors can send non-broadcast messages either through their internal ports of their connection or through their external ports. A seeking computer uses external ports when locating a portal computer.

In one embodiment, the broadcast technique establishes the computer connections using the TCP/IP communications protocol, which is a point-to-point protocol, as the underlying network. The TCP/IP protocol provides for reliable and ordered delivery of messages between computers. The TCP/IP protocol provides each computer with a "port space" that is shared among all the processes that may execute on that computer. The ports are identified by numbers from 0 to 65,535. The first 2056 ports are reserved for specific applications (e.g., port 80 for HTTP messages). The remainder of the ports are user ports that are available to any process. In one embodiment, a set of port numbers can be reserved for use by the computer connected to the broadcast channel. In an alternative embodiment, the port numbers used are dynamically identified by each computer. Each computer dynamically identifies an available port to be used as its call-in port. This call-in port is used to establish connections with the external port and the internal ports. Each computer that is connected to the broadcast channel can receive

non-broadcast messages through its external port. A seeking computer tries "dialing" the port numbers of the portal computers until a portal computer "answers," a call on its call-in port. A portal computer answers when it is connected to or attempting to connect to the broadcast channel and its call-in port is dialed. (In this description, a telephone metaphor is used to describe the connections.) When a computer receives a call on its call-in port, it transfers the call to another port. Thus, the seeking computer actually communicates through that transfer-to port, which is the external port. The call is transferred so that other computers can place calls to that computer via the call-in port. The seeking computer then communicates via that external port to request the portal computer to assist in connecting the seeking computer to the broadcast channel. The seeking computer could identify the call-in port number of a portal computer by successively dialing each port in port number order. As discussed below in detail, the broadcast technique uses a hashing algorithm to select the port number order, which may result in improved performance.

A seeking computer could connect to the broadcast channel by connecting to computers either directly connected to the found portal computer or directly connected to one of its neighbors. A possible problem with such a scheme for identifying the neighbors for the seeking computer is that the diameter of the broadcast channel may increase when each seeking computer uses the same found portal computer and establishes a connection to the broadcast channel directly through that found portal computer. Conceptually, the graph becomes elongated in the direction of where the new nodes are added. FIGS. 4A-4C illustrate that possible problem. FIG. 4A illustrates the broadcast channel of FIG. 1 with an added computer. Computer J was connected to the broadcast channel by edge pinning edges C-D and E-H to computer J. The diameter of this broadcast channel is still two. FIG. 4B illustrates the broadcast channel of FIG. 4A with an added computer. Computer K was connected to the broadcast channel by edge pinning edges E-J and B-C to computer K. The diameter of this broadcast channel is three, because the shortest path from computer G to computer K is through edges G-A, A-E, and E-K. FIG. 4C also illustrates the broadcast channel of FIG. 4A with an added computer. Computer K was connected to the broadcast channel by edge pinning edges D-G and E-J to computer K. The diameter of this broadcast channel is, however, still two. Thus, the selection of neighbors impacts the diameter of the broadcast channel. To help minimize the diameter, the broadcast technique uses a random selection technique to identify the four neighbors of a computer in the seeking connection state. The random selection technique tends to distribute the connections to new seeking computers throughout the computers of the broadcast channel which may result in smaller overall diameters.

Broadcasting Through the Graph

As described above, each computer that is connected to the broadcast channel can broadcast messages onto the broadcast channel and does receive all messages that are broadcast on the broadcast channel. The computer that originates a message to be broadcast sends that message to each of its four neighbors using the internal connections. When a computer receives a broadcast message from a neighbor, it sends the message to its three other neighbors. Each computer on the broadcast channel, except the originating computer, will thus receive a copy of each broadcast message from each of its four neighbors. Each computer, however, only sends the first copy of the message that it

receives to its neighbors and disregards subsequently received copies. Thus, the total number of copies of a message that is sent between the computers is $3N+1$, where N is the number of computers connected to the broadcast channel. Each computer sends three copies of the message, except for the originating computer, which sends four copies of the message.

The redundancy of the message sending helps to ensure the overall reliability of the broadcast channel. Since each computer has four connections to the broadcast channel, if one computer fails during the broadcast of a message, its neighbors have three other connections through which they will receive copies of the broadcast message. Also, if the internal connection between two computers is slow, each computer has three other connections through which it may receive a copy of each message sooner.

Each computer that originates a message numbers its own messages sequentially. Because of the dynamic nature of the broadcast channel and because there are many possible connection paths between computers, the messages may be received out of order. For example, the distance between an originating computer and a certain receiving computer may be four. After sending the first message, the originating computer and receiving computer may become neighbors and thus the distance between them changes to one. The first message may have to travel a distance of four to reach the receiving computer. The second message only has to travel a distance of one. Thus, it is possible for the second message to reach the receiving computer before the first message.

When the broadcast channel is in a steady state (i.e., no computers connecting or disconnecting from the broadcast channel), out-of-order messages are not a problem because each computer will eventually receive both messages and can queue messages until all earlier ordered messages are received. If, however, the broadcast channel is not in a steady state, then problems can occur. In particular, a computer may connect to the broadcast channel after the second message has already been received and forwarded on by its new neighbors. When a new neighbor eventually receives the first message, it sends the message to the newly connected computer. Thus, the newly connected computer will receive the first message, but will not receive the second message. If the newly connected computer needs to process the messages in order, it would wait indefinitely for the second message.

One solution to this problem is to have each computer queue all the messages that it receives until it can send them in their proper order to its neighbors. This solution, however, may tend to slow down the propagation of messages through the computers of the broadcast channel. Another solution that may have less impact on the propagation speed is to queue messages only at computers who are neighbors of the newly connected computers. Each already connected neighbor would forward messages as it receives them to its other neighbors who are not newly connected, but not to the newly connected neighbor. The already connected neighbor would only forward messages from each originating computer to the newly connected computer when it can ensure that no gaps in the messages from that originating computer will occur. In one embodiment, the already connected neighbor may track the highest sequence number of the messages already received and forwarded on from each originating computer. The already connected computer will send only higher numbered messages from the originating computers to the newly connected computer. Once all lower numbered messages have been received from all originating computers, then the already connected computer can treat

the newly connected computer as its other neighbors and simply forward each message as it is received. In another embodiment, each computer may queue messages and only forwards to the newly connected computer those messages as the gaps are filled in. For example, a computer might receive messages 4 and 5 and then receive message 3. In such a case, the already connected computer would forward queue messages 4 and 5. When message 3 is finally received, the already connected computer will send messages 3, 4, and 5 to the newly connected computer. If messages 4 and 5 were sent to the newly connected computer before message 3, then the newly connected computer would process messages 4 and 5 and disregard message 3. Because the already connected computer queues messages 4 and 5, the newly connected computer will be able to process message 3. It is possible that a newly connected computer will receive a set of messages from an originating computer through one neighbor and then receive another set of message from the same originating computer through another neighbor. If the second set of messages contains a message that is ordered earlier than the messages of the first set received, then the newly connected computer may ignore that earlier ordered message if the computer already processed those later ordered messages.

Decomposing the Graph

A connected computer disconnects from the broadcast channel either in a planned or unplanned manner. When a computer disconnects in a planned manner, it sends a disconnect message to each of its four neighbors. The disconnect message includes a list that identifies the four neighbors of the disconnecting computer. When a neighbor receives the disconnect message, it tries to connect to one of the computers on the list. In one embodiment, the first computer in the list will try to connect to the second computer in the list, and the third computer in the list will try to connect to the fourth computer in the list. If a computer cannot connect (e.g. the first and second computers are already connected), then the computers may try connecting in various other combinations. If connections cannot be established, each computer broadcasts a message that it needs to establish a connection with another computer. When a computer with an available internal port receives the message, it can then establish a connection with the computer that broadcast the message. FIGS. 5A-5D illustrate the disconnecting of a computer from the broadcast channel. FIG. 5A illustrates the disconnecting of a computer from the broadcast channel in a planned manner. When computer H decides to disconnect, it sends its list of neighbors to each of its neighbors (computers A, E, F and I) and then disconnects from each of its neighbors. When computers A and I receive the message they establish a connection between them as indicated by the dashed line, and similarly for computers E and F.

When a computer disconnects in an unplanned manner, such as resulting from a power failure, the neighbors connected to the disconnected computer recognize the disconnection when each attempts to send its next message to the now disconnected computer. Each former neighbor of the disconnected computer recognizes that it is short one connection (i.e., it has a hole or empty port). When a connected computer detects that one of its neighbors is now disconnected, it broadcasts a port connection request on the broadcast channel, which indicates that it has one internal port that needs a connection. The port connection request identifies the call-in port of the requesting computer. When a connected computer that is also short a connection receives

the connection request, it communicates with the requesting computer through its external port to establish a connection between the two computers. FIG. 5B illustrates the disconnecting of a computer from the broadcast channel in an unplanned manner. In this illustration, computer H has disconnected in an unplanned manner. When each of its neighbors, computers A, E, F, and I, recognizes the disconnection, each neighbor broadcasts a port connection request indicating that it needs to fill an empty port. As shown by the dashed lines, computers F and I and computers A and E respond to each other's requests and establish a connection.

It is possible that a planned or unplanned disconnection may result in two neighbors each having an empty internal port. In such a case, since they are neighbors, they are already connected and cannot fill their empty ports by connecting to each other. Such a condition is referred to as the "neighbors with empty ports" condition. Each neighbor broadcasts a port connection request when it detects that it has an empty port as described above. When a neighbor receives the port connection request from the other neighbor, it will recognize the condition that its neighbor also has an empty port. Such a condition may also occur when the broadcast channel is in the small regime. The condition can only be corrected when in the large regime. When in the small regime, each computer will have less than four neighbors. To detect this condition in the large regime, which would be a problem if not repaired, the first neighbor to receive the port connection request recognizes the condition and sends a condition check message to the other neighbor. The condition check message includes a list of the neighbors of the sending computer. When the receiving computer receives the list, it compares the list to its own list of neighbors. If the lists are different, then this condition has occurred in the large regime and repair is needed. To repair this condition, the receiving computer will send a condition repair request to one of the neighbors of the sending computer which is not already a neighbor of the receiving computer. When the computer receives the condition repair request, it disconnects from one of its neighbors (other than the neighbor that is involved with the condition) and connects to the computer that sent the condition repair request. Thus, one of the original neighbors involved in the condition will have had a port filled. However, two computers are still in need of a connection, the other original neighbor and the computer that is now disconnected from the computer that received the condition repair request. Those two computers send out port connection requests. If those two computers are not neighbors, then they will connect to each other when they receive the requests. If, however, the two computers are neighbors, then they repeat the condition repair process until two non-neighbors are in need of connections.

It is possible that the two original neighbors with the condition may have the same set of neighbors. When the neighbor that receives the condition check message determines that the sets of neighbors are the same, it sends a condition double check message to one of its neighbors other than the neighbor who also has the condition. When the computer receives the condition double check message, it determines whether it has the same set of neighbors as the sending computer. If so, the broadcast channel is in the small regime and the condition is not a problem. If the set of neighbors are different, then the computer that received the condition double check message sends a condition check message to the original neighbors with the condition. The computer that receives that condition check message directs one of its neighbors to connect to one of the original

neighbors with the condition by sending a condition repair message. Thus, one of the original neighbors with the condition will have its port filled.

FIG. 5C illustrates the neighbors with empty ports condition. In this illustration, computer H disconnected in an unplanned manner, but computers F and I responded to the port connection request of the other and are now connected together. The other former neighbors of computer H, computers A and E, are already neighbors, which gives rise to the neighbors with empty ports condition. In this example, computer E received the port connection request from computer A, recognized the possible condition, and sent (since they are neighbors via the internal connection) a condition check message with a list of its neighbors to computer A. When computer A received the list, it recognized that computer E has a different set of neighbor (i.e., the broadcast channel is in the large regime). Computer A selected computer D, which is a neighbor of computer E and sent it a condition repair request. When computer D received the condition repair request, it disconnected from one of its neighbors (other than computer E), which is computer G in this example. Computer D then connected to computer A. FIG. 5D illustrates two computers that are not neighbors who now have empty ports. Computers E and G now have empty ports and are not currently neighbors. Therefore, computers E and G can connect to each other.

FIGS. 5E and 5F further illustrate the neighbors with empty ports condition. FIG. 5E illustrates the neighbors with empty ports condition in the small regime. In this example, if computer E disconnected in an unplanned manner, then each computer broadcasts a port connection request when it detects the disconnect. When computer A receives the port connection request from computer B, it detects the neighbors with empty ports condition and sends a condition check message to computer B. Computer B recognizes that it has the same set of neighbors (computer C and D) as computer A and then sends a condition double check message to computer C. Computer C recognizes that the broadcast channel is in the small regime because it also has the same set of neighbors as computers A and B, computer C may then broadcast a message indicating that the broadcast channel is in the small regime.

FIG. 5F illustrates the situation of FIG. 5E when in the large regime. As discussed above, computer C receives the condition double check message from computer B. In this case, computer C recognizes that the broadcast channel is in the large regime because it has a set of neighbors that is different from computer B. The edges extending up from computer C and D indicate connections to other computers. Computer C then sends a condition check message to computer B. When computer B receives the condition check message, it sends a condition repair message to one of the neighbors of computer C. The computer that receives the condition repair message disconnects from one of its neighbors, other than computer C, and tries to connect to computer B and the neighbor from which it disconnected tries to connect to computer A.

Port Selection

As described above, the TCP/IP protocol designates ports above number 2056 as user ports. The broadcast technique uses five user port numbers on each computer: one external port and four internal ports. Generally, user ports cannot be statically allocated to an application program because other applications programs executing on the same computer may use conflicting port numbers. As a result, in one

embodiment, the computers connected to the broadcast channel dynamically allocate their port numbers. Each computer could simply try to locate the lowest number unused port on that computer and use that port as the call-in port. A seeking computer, however, does not know in advance the call-in port number of the portal computers when the port numbers are dynamically allocated. Thus, a seeking computer needs to dial ports of a portal computer starting with the lowest port number when locating the call-in port of a portal computer. If the portal computer is connected to (or attempting to connect to) the broadcast channel, then the seeking computer would eventually find the call-in port. If the portal computer is not connected, then the seeking computer would eventually dial every user port. In addition, if each application program on a computer tried to allocate low-ordered port numbers, then a portal computer may end up with a high-numbered port for its call-in port because many of the low-ordered port numbers would be used by other application programs. Since the dialing of a port is a relatively slow process, it would take the seeking computer a long time to locate the call-in port of a portal computer. To minimize this time, the broadcast technique uses a port ordering algorithm to identify the port number order that a portal computer should use when finding an available port for its call-in port. In one embodiment, the broadcast technique uses a hashing algorithm to identify the port order. The algorithm preferably distributes the ordering of the port numbers randomly through out the user port number space and only selects each port number once. In addition, every time the algorithm is executed on any computer for a given channel type and channel instance, it generates the same port ordering. As described below, it is possible for a computer to be connected to multiple broadcast channels that are uniquely identified by channel type and channel instance. The algorithm may be "seeded" with channel type and channel instance in order to generate a unique ordering of port numbers for each broadcast channel. Thus, a seeking computer will dial the ports of a portal computer in the same order as the portal computer used when allocating its call-in port.

If many computers are at the same time seeking connection to a broadcast channel through a single portal computer, then the ports of the portal computer may be busy when called by seeking computers. The seeking computers would typically need to keep on redialing a busy port. The process of locating a call-in port may be significantly slowed by such redialing. In one embodiment, each seeking computer may each reorder the first few port numbers generated by the hashing algorithm. For example, each seeking computer could randomly reorder the first eight port numbers generated by the hashing algorithm. The random ordering could also be weighted where the first port number generated by the hashing algorithm would have a 50% chance of being first in the reordering, the second port number would have a 25% chance of being first in the reordering, and so on. Because the seeking computers would use different orderings, the likelihood of finding a busy port is reduced. For example, if the first eight port numbers are randomly selected, then it is possible that eight seeking computers could be simultaneously dialing ports in different sequences which would reduce the chances of dialing a busy port.

Locating a Portal Computer

Each computer that can connect to the broadcast channel has a list of one or more portal computers through which it can connect to the broadcast channel. In one embodiment, each computer has the same set of portal computers. A

seeking computer locates a portal computer that is connected to the broadcast channel by successively dialing the ports of each portal computer in the order specified by an algorithm. A seeking computer could select the first portal computer and then dial all its ports until a call-in port of a computer that is fully connected to the broadcast channel is found. If no call-in port is found, then the seeking computer would select the next portal computer and repeat the process until a portal computer with such a call-in port is found. A problem with such a seeking technique is that all user ports of each portal computer are dialed until a portal computer fully connected to the broadcast channel is found. In an alternate embodiment, the seeking computer selects a port number according to the algorithm and then dials each portal computer at that port number. If no acceptable call-in port to the broadcast channel is found, then the seeking computer selects the next port number and repeats the process. Since the call-in ports are likely allocated at lower-ordered port numbers, the seeking computer first dials the port numbers that are most likely to be call-in ports of the broadcast channel. The seeking computers may have a maximum search depth, that is the number of ports that it will dial when seeking a portal computer that is fully connected. If the seeking computer exhausts its search depth, then either the broadcast channel has not yet been established or, if the seeking computer is also a portal computer, it can then establish the broadcast channel with itself as the first fully connected computer.

When a seeking computer locates a portal computer that is itself not fully connected, the two computers do not connect when they first locate each other because the broadcast channel may already be established and accessible through a higher-ordered port number on another portal computer. If the two seeking computers were to connect to each other, then two disjoint broadcast channels would be formed. Each seeking computer can share its experience in trying to locate a portal computer with the other seeking computer. In particular, if one seeking computer has searched all the portal computers to a depth of eight, then the one seeking computer can share that it has searched to a depth of eight with another seeking computer. If that other seeking computer has searched to a depth of, for example, only four, it can skip searching through depths five through eight and that other seeking computer can advance its searching to a depth of nine.

In one embodiment, each computer may have a different set of portal computers and a different maximum search depth. In such a situation, it may be possible that two disjoint broadcast channels are formed because a seeking computer cannot locate a fully connected port computer at a higher depth. Similarly, if the set of portal computers are disjoint, then two separate broadcast channels would be formed.

Identifying Neighbors for a Seeking Computer

As described above, the neighbors of a newly connecting computer are preferably selected randomly from the set of currently connected computers. One advantage of the broadcast channel, however, is that no computer has global knowledge of the broadcast channel. Rather, each computer has local knowledge of itself and its neighbors. This limited local knowledge has the advantage that all the connected computers are peers (as far as the broadcasting is concerned) and the failure of any one computer (actually any three computers when in the 4-regular and 4-connect form) will not cause the broadcast channel to fail. This local knowledge makes it difficult for a portal computer to randomly select four neighbors for a seeking computer.

To select the four computers, a portal computer sends an edge connection request message through one of its internal connections that is randomly selected. The receiving computer again sends the edge connection request message through one of its internal connections that is randomly selected. This sending of the message corresponds to a random walk through the graph that represents the broadcast channel. Eventually, a receiving computer will decide that the message has traveled far enough to represent a randomly selected computer. That receiving computer will offer the internal connection upon which it received the edge connection request message to the seeking computer for edge pinning. Of course, if either of the computers at the end of the offered internal connection are already neighbors of the seeking computer, then the seeking computer cannot connect through that internal connection. The computer that decided that the message has traveled far enough will detect this condition of already being a neighbor and send the message to a randomly selected neighbor.

In one embodiment, the distance that the edge connection request message travels is established by the portal computer to be approximately twice the estimated diameter of the broadcast channel. The message includes an indication of the distance that it is to travel. Each receiving computer decrements that distance to travel before sending the message on. The computer that receives a message with a distance to travel that is zero is considered to be the randomly selected computer. If that randomly selected computer cannot connect to the seeking computer (e.g., because it is already connected to it), then that randomly selected computer forwards the edge connection request to one of its neighbors with a new distance to travel. In one embodiment, the forwarding computer toggles the new distance to travel between zero and one to help prevent two computers from sending the message back and forth between each other.

Because of the local nature of the information maintained by each computer connected to the broadcast channel, the computers need not generally be aware of the diameter of the broadcast channel. In one embodiment, each message sent through the broadcast channel has a distance traveled field. Each computer that forwards a message increments the distance traveled field. Each computer also maintains an estimated diameter of the broadcast channel. When a computer receives a message that has traveled a distance that indicates that the estimated diameter is too small, it updates its estimated diameter and broadcasts an estimated diameter message. When a computer receives an estimated diameter message that indicates a diameter that is larger than its own estimated diameter, it updates its own estimated diameter. This estimated diameter is used to establish the distance that an edge connection request message should travel.

External Data Representation

The computers connected to the broadcast channel may internally store their data in different formats. For example, one computer may use 32-bit integers, and another computer may use 64-bit integers. As another example, one computer may use ASCII to represent text and another computer may use Unicode. To allow communications between heterogeneous computers, the messages sent over the broadcast channel may use the XDR ("external Data Representation") format.

The underlying peer-to-peer communications protocol may send multiple messages in a single message stream. The traditional technique for retrieving messages from a stream has been to repeatedly invoke an operating system routine to

15

retrieve the next message in the stream. The retrieval of each message may require two calls to the operating system: one to retrieve the size of the next message and the other to retrieve the number of bytes indicated by the retrieved size. Such calls to the operating system can, however, be very slow in comparison to the invocations of local routines. To overcome the inefficiencies of such repeated calls, the broadcast technique in one embodiment, uses XDR to identify the message boundaries in a stream of messages. The broadcast technique may request the operating system to provide the next, for example, 1,024 bytes from the stream. The broadcast technique can then repeatedly invoke the XDR routines to retrieve the messages and use the success or failure of each invocation to determine whether another block of 1,024 bytes needs to be retrieved from the operating system. The invocation of XDR routines do not involve system calls and are thus more efficient than repeated system calls.

M-Regular

In the embodiment described above, each fully connected computer has four internal connections. The broadcast technique can be used with other numbers of internal connections. For example, each computer could have 6, 8, or any even number of internal connections. As the number of internal connections increase, the diameter of the broadcast channel tends to decrease, and thus propagation time for a message tends to decrease. The time that it takes to connect a seeking computer to the broadcast channel may, however, increase as the number of internal connections increases. When the number of internal connectors is even, then the broadcast channel can be maintained as m-regular and m-connected (in the steady state). If the number of internal connections is odd, then when the broadcast channel has an odd number of computers connected, one of the computers will have less than that odd number of internal connections. In such a situation, the broadcast network is neither m-regular nor m-connected. When the next computer connects to the broadcast channel, it can again become m-regular and m-connected. Thus, with an odd number of internal connections, the broadcast channel toggles between being and not being m-regular and m-connected.

Components

FIG. 6 is a block diagram illustrating components of a computer that is connected to a broadcast channel. The above description generally assumed that there was only one broadcast channel and that each computer had only one connection to that broadcast channel. More generally, a network of computers may have multiple broadcast channels, each computer may be connected to more than one broadcast channel, and each computer can have multiple connections to the same broadcast channel. The broadcast channel is well suited for computer processes (e.g., application programs) that execute collaboratively, such as network meeting programs. Each computer process can connect to one or more broadcast channels. The broadcast channels can be identified by channel type (e.g., application program name) and channel instance that represents separate broadcast channels for that channel type. When a process attempts to connect to a broadcast channel, it seeks a process currently connected to that broadcast channel that is executing on a portal computer. The seeking process identifies the broadcast channel by channel type and channel instance.

Computer 600 includes multiple application programs 601 executing as separate processes. Each application program interfaces with a broadcaster component 602 for each

16

broadcast channel to which it is connected. The broadcaster component may be implemented as an object that is instantiated within the process space of the application program. Alternatively, the broadcaster component may execute as a separate process or thread from the application program. In one embodiment, the broadcaster component provides functions (e.g., methods of class) that can be invoked by the application programs. The primary functions provided may include a connect function that an application program invokes passing an indication of the broadcast channel to which the application program wants to connect. The application program may provide a callback routine that the broadcaster component invokes to notify the application program that the connection has been completed, that is the process enters the fully connected state. The broadcaster component may also provide an acquire message function that the application program can invoke to retrieve the next message that is broadcast on the broadcast channel. Alternatively, the application program may provide a callback routine (which may be a virtual function provided by the application program) that the broadcaster component invokes to notify the application program that a broadcast message has been received. Each broadcaster component allocates a call-in port using the hashing algorithm. When calls are answered at the call-in port, they are transferred to other ports that serve as the external and internal ports.

The computers connecting to the broadcast channel may include a central processing unit, memory, input devices (e.g., keyboard and pointing device), output devices (e.g., display devices), and storage devices (e.g., disk drives). The memory and storage devices are computer-readable medium that may contain computer instructions that implement the broadcaster component. In addition, the data structures and message structures may be stored or transmitted via a signal transmitted on a computer-readable media, such as a communications link.

FIG. 7 is a block diagram illustrating the sub-components of the broadcaster component in one embodiment. The broadcaster component includes a connect component 701, an external dispatcher 702, an internal dispatcher 703 for each internal connection, an acquire message component 704 and a broadcast component 712. The application program may provide a connect callback component 710 and a receive response component 711 that are invoked by the broadcaster component. The application program invokes the connect component to establish a connection to a designated broadcast channel. The connect component identifies the external port and installs the external dispatcher for handling messages that are received on the external port. The connect component invokes the seek portal computer component 705 to identify a portal computer that is connected to the broadcast channel and invokes the connect request component 706 to ask the portal computer (if fully connected) to select neighbor processes for the newly connecting process. The external dispatcher receives external messages, identifies the type of message, and invokes the appropriate handling routine 707. The internal dispatcher receives the internal messages, identifies the type of message, and invokes the appropriate handling routine 708. The received broadcast messages are stored in the broadcast message queue 709. The acquire message component is invoked to retrieve messages from the broadcast queue. The broadcast component is invoked by the application program to broadcast messages in the broadcast channel.

The following tables list messages sent by the broadcaster components.

External Messages

<u>EXTERNAL MESSAGES</u>	
Message Type	Description
seeking_	Indicates that a seeking process would like to know
connection__call	whether the receiving process is fully connected to the broadcast channel
connection__	Indicates that the sending process would like the
request__call	receiving process to initiate a connection of the sending process to the broadcast channel
edge__proposal__	Indicates that the sending process is proposing an edge
call	through which the receiving process can connect to the broadcast channel (i.e., edge pinning)
port__	Indicates that the sending process is proposing a port
connection__call	through which the receiving process can connect to the broadcast channel
connected__stmt	Indicates that the sending process is connected to the broadcast channel
condition__	Indicates that the receiving process should disconnect
repair__stmt	from one of its neighbors and connect to one of the processes involved in the neighbors with empty port condition

Internal Messages

<u>INTERNAL MESSAGES</u>	
Message Type	Description
broadcast__stmt	Indicates a message that is being broadcast through the broadcast channel for the application programs
connection__port__	Indicates that the designated process is looking for a
search__stmt	port through which it can connect to the broadcast channel
connection__edge__	Indicates that the requesting process is looking for
search__call	an edge through which it can connect to the broadcast channel
connection__edge__	Indicates whether the edge between this process and
search__resp	the sending neighbor has been accepted by the requesting party
diameter__estimate__	Indicates an estimated diameter of the broadcast
stmt	channel
diameter__reset__	Indicates to reset the estimated diameter to
stmt	indicated diameter
disconnect__stmt	Indicates that the sending neighbor is disconnecting
	from the broadcast channel
condition__check__	Indicates that neighbors with empty port condition
stmt	have been detected
condition__double__	Indicates that the neighbors with empty ports have
check__stmt	the same set of neighbors
shutdown__stmt	Indicates that the broadcast channel is being
	shutdown

Flow Diagrams

FIGS. 8-34 are flow diagrams illustrating the processing of the broadcaster component in one embodiment. FIG. 8 is a flow diagram illustrating the processing of the connect routine in one embodiment. This routine is passed a channel type (e.g., application name) and channel instance (e.g., session identifier), that identifies the broadcast channel to which this process wants to connect. The routine is also passed auxiliary information that includes the list of portal computers and a connection callback routine. When the connection is established, the connection callback routine is invoked to notify the application program. When this process invokes this routine, it is in the seeking connection state. When a portal computer is located that is connected and this routine connects to at least one neighbor, this

process enters the partially connected state, and when the process eventually connects to four neighbors, it enters the fully connected state. When in the small regime, a fully connected process may have less than four neighbors. In block 801, the routine opens the call-in port through which the process is to communicate with other processes when establishing external and internal connections. The port is selected as the first available port using the hashing algorithm described above. In block 802, the routine sets the connect time to the current time. The connect time is used to identify the instance of the process that is connected through this external port. One process may connect to a broadcast channel of a certain channel type and channel instance using one call-in port and then disconnects, and another process may then connect to that same broadcast channel using the same call-in port. Before the other process becomes fully connected, another process may try to communicate with it thinking it is the fully connected old process. In such a case, the connect time can be used to identify this situation. In block 803, the routine invokes the seek portal computer routine passing the channel type and channel instance. The seek portal computer routine attempts to locate a portal computer through which this process can connect to the broadcast channel for the passed type and instance. In decision block 804, if the seek portal computer routine is successful in locating a fully connected process on that portal computer, then the routine continues at block 805, else the routine returns an unsuccessful indication. In decision block 805, if no portal computer other than the portal computer on which the process is executing was located, then this is the first process to fully connect to broadcast channel and the routine continues at block 806, else the routine continues at block 808. In block 806, the routine invokes the achieve connection routine to change the state of this process to fully connected. In block 807, the routine installs the external dispatcher for processing messages received through this process' external port for the passed channel type and channel instance. When a message is received through that external port, the external dispatcher is invoked. The routine then returns. In block 808, the routine installs an external dispatcher. In block 809, the routine invokes the connect request routine to initiate the process of identifying neighbors for the seeking computer. The routine then returns.

FIG. 9 is a flow diagram illustrating the processing of the seek portal computer routine in one embodiment. This routine is passed the channel type and channel instance of the broadcast channel to which this process wishes to connect. This routine, for each search depth (e.g., port number), checks the portal computers at that search depth. If a portal computer is located at that search depth with a process that is fully connected to the broadcast channel, then the routine returns an indication of success. In blocks 902-911, the routine loops selecting each search depth until a process is located. In block 902, the routine selects the next search depth using a port number ordering algorithm. In decision block 903, if all the search depths have already been selected during this execution of the loop, that is for the currently selected depth, then the routine returns a failure indication, else the routine continues at block 904. In blocks 904-911, the routine loops selecting each portal computer and determining whether a process of that portal computer is connected to (or attempting to connect to) the broadcast channel with the passed channel type and channel instance. In block 904, the routine selects the next portal computer. In decision block 905, if all the portal computers have already been selected, then the routine loops to block 902 to select

19

the next search depth, else the routine continues at block 906. In block 906, the routine dials the selected portal computer through the port represented by the search depth. In decision block 30 907, if the dialing was successful, then the routine continues at block 908, else the routine loops to block 904 to select the next portal computer. The dialing will be successful if the dialed port is the call-in port of the broadcast channel of the passed channel type and channel instance of a process executing on that portal computer. In block 908, the routine invokes a contact process routine, which contacts the answering process of the portal computer through the dialed port and determines whether that process is fully connected to the broadcast channel. In block 909, the routine hangs up on the selected portal computer. In decision block 910, if the answering process is fully connected to the broadcast channel, then the routine returns a success indicator, else the routine continues at block 911. In block 911, the routine invokes the check for external call routine to determine whether an external call has been made to this process as a portal computer and processes that call. The routine then loops to block 904 to select the next portal computer.

FIG. 10 is a flow diagram illustrating the processing of the contact process routine in one embodiment. This routine determines whether the process of the selected portal computer that answered the call-in to the selected port is fully connected to the broadcast channel. In block 1001, the routine sends an external message (i.e., seeking_connection_call) to the answering process indicating that a seeking process wants to know whether the answering process is fully connected to the broadcast channel. In block 1002, the routine receives the external response message from the answering process. In decision block 1003, if the external response message is successfully received (i.e., seeking_connection_resp), then the routine continues at block 1004, else the routine returns. Wherever the broadcast component requests to receive an external message, it sets a time out period. If the external message is not received within that time out period, the broadcaster component checks its own call-in port to see if another process is calling it. In particular, the dialed process may be calling the dialing process, which may result in a deadlock situation. The broadcaster component may repeat the receive request several times. If the expected message is not received, then the broadcaster component handles the error as appropriate. In decision block 1004, if the answering process indicates in its response message that it is fully connected to the broadcast channel, then the routine continues at block 1005, else the routine continues at block 1006. In block 1005, the routine adds the selected portal computer to a list of connected portal computers and then returns. In block 1006, the routine adds the answering process to a list of fellow seeking processes and then returns.

FIG. 11 is a flow diagram illustrating the processing of the connect request routine in one embodiment. This routine requests a process of a portal computer that was identified as being fully connected to the broadcast channel to initiate the connection of this process to the broadcast channel. In decision block 1101, if at least one process of a portal computer was located that is fully connected to the broadcast channel, then the routine continues at block 1103, else the routine continues at block 1102. A process of the portal computer may no longer be in the list if it recently disconnected from the broadcast channel. In one embodiment, a seeking computer may always search its entire search depth and find multiple portal computers through which it can connect to the broadcast channel. In block 1102, the routine

20

restarts the process of connecting to the broadcast channel and returns. In block 1103, the routine dials the process of one of the found portal computers through the call-in port. In decision block 1104, if the dialing is successful, then the routine continues at block 1105, else the routine continues at block 1113. The dialing may be unsuccessful if, for example, the dialed process recently disconnected from the broadcast channel. In block 1105, the routine sends an external message to the dialed process requesting a connection to the broadcast channel (i.e., connection_request_call). In block 1106, the routine receives the response message (i.e., connection_request_resp). In decision block 1107, if the response message is successfully received, then the routine continues at block 1108, else the routine continues at block 1113. In block 1108, the routine sets the expected number of holes (i.e., empty internal connections) for this process based on the received response. When in the large regime, the expected number of holes is zero. When in the small regime, the expected number of holes varies from one to three. In block 1109, the routine sets the estimated diameter of the broadcast channel based on the received response. In decision block 1111, if the dialed process is ready to connect to this process as indicated by the response message, then the routine continues at block 1112, else the routine continues at block 1113. In block 1112, the routine invokes the add neighbor routine to add the answering process as a neighbor to this process. This adding of the answering process typically occurs when the broadcast channel is in the small regime. When in the large regime, the random walk search for a neighbor is performed. In block 1113, the routine hangs up the external connection with the answering process computer and then returns.

FIG. 12 is a flow diagram of the processing of the check for external call routine in one embodiment. This routine is invoked to identify whether a fellow seeking process is attempting to establish a connection to the broadcast channel through this process. In block 1201, the routine attempts to answer a call on the call-in port. In decision block 1202, if the answer is successful, then the routine continues at block 1203, else the routine returns. In block 1203, the routine receives the external message from the external port. In decision block 1204, if the type of the message indicates that a seeking process is calling (i.e., seeking_connection_call), then the routine continues at block 1205, else the routine returns. In block 1205, the routine sends an external message (i.e., seeking_connection_resp) to the other seeking process indicating that this process is also seeking a connection. In decision block 1206, if the sending of the external message is successful, then the routine continues at block 1207, else the routine returns. In block 1207, the routine adds the other seeking process to a list of fellow seeking processes and then returns. This list may be used if this process can find no process that is fully connected to the broadcast channel. In which case, this process may check to see if any fellow seeking process were successful in connecting to the broadcast channel. For example, a fellow seeking process may become the first process fully connected to the broadcast channel.

FIG. 13 is a flow diagram of the processing of the achieve connection routine in one embodiment. This routine sets the state of this process to fully connected to the broadcast channel and invokes a callback routine to notify the application program that the process is now fully connected to the requested broadcast channel. In block 1301, the routine sets the connection state of this process to fully connected. In block 1302, the routine notifies fellow seeking processes that it is fully connected by sending a connected external

message to them (i.e., `connected_stmt`). In block 1303, the routine invokes the connect callback routine to notify the application program and then returns.

FIG. 14 is a flow diagram illustrating the processing of the external dispatcher routine in one embodiment. This routine is invoked when the external port receives a message. This routine retrieves the message, identifies the external message type, and invokes the appropriate routine to handle that message. This routine loops processing each message until all the received messages have been handled. In block 1401, the routine answers (e.g., picks up) the external port and retrieves an external message. In decision block 1402, if a message was retrieved, then the routine continues at block 1403, else the routine hangs up on the external port in block 1415 and returns. In decision block 1403, if the message type is for a process seeking a connection (i.e., `seeking_connection_call`), then the routine invokes the handle seeking connection call routine in block 1404, else the routine continues at block 1405. In decision block 1405, if the message type is for a connection request call (i.e., `connection_request_call`), then the routine invokes the handle connection request call routine in block 1406, else the routine continues at block 1407. In decision block 1407, if the message type is edge proposal call (i.e., `edge_proposal_call`), then the routine invokes the handle edge proposal call routine in block 1408, else the routine continues at block 1409. In decision block 1409, if the message type is port connect call (i.e., `port_connect_call`), then the routine invokes the handle port connection call routine in block 1410, else the routine continues at block 1411. In decision block 1411, if the message type is a connected statement (i.e., `connected_stmt`), the routine invokes the handle connected statement in block 1412, else the routine continues at block 1212. In decision block 1412, if the message type is a condition repair statement (i.e., `condition_repair_stmt`), then the routine invokes the handle condition repair routine in block 1413, else the routine loops to block 1414 to process the next message. After each handling routine is invoked, the routine loops to block 1414. In block 1414, the routine hangs up on the external port and continues at block 1401 to receive the next message.

FIG. 15 is a flow diagram illustrating the processing of the handle seeking connection call routine in one embodiment. This routine is invoked when a seeking process is calling to identify a portal computer through which it can connect to the broadcast channel. In decision block 1501, if this process is currently fully connected to the broadcast channel identified in the message, then the routine continues at block 1502, else the routine continues at block 1503. In block 1502, the routine sets a message to indicate that this process is fully connected to the broadcast channel and continues at block 1505. In block 1503, the routine sets a message to indicate that this process is not fully connected. In block 1504, the routine adds the identification of the seeking process to a list of fellow seeking processes. If this process is not fully connected, then it is attempting to connect to the broadcast channel. In block 1505, the routine sends the external message response (i.e., `seeking_connection_resp`) to the seeking process and then returns.

FIG. 16 is a flow diagram illustrating processing of the handle connection request call routine in one embodiment. This routine is invoked when the calling process wants this process to initiate the connection of the process to the broadcast channel. This routine either allows the calling process to establish an internal connection with this process (e.g., if in the small regime) or starts the process of identifying a process to which the calling process can connect. In

decision block 1601, if this process is currently fully connected to the broadcast channel, then the routine continues at block 1603, else the routine hangs up on the external port in block 1602 and returns. In block 1603, the routine sets the number of holes that the calling process should expect in the response message. In block 1604, the routine sets the estimated diameter in the response message. In block 1605, the routine indicates whether this process is ready to connect to the calling process. This process is ready to connect when the number of its holes is greater than zero and the calling process is not a neighbor of this process. In block 1606, the routine sends to the calling process an external message that is responsive to the connection request call (i.e., `connection_request_resp`). In block 1607, the routine notes the number of holes that the calling process needs to fill as indicated in the request message. In decision block 1608, if this process is ready to connect to the calling process, then the routine continues at block 1609, else the routine continues at block 1611. In block 1609, the routine invokes the add neighbor routine to add the calling process as a neighbor. In block 1610, the routine decrements the number of holes that the calling process needs to fill and continues at block 1611. In block 1611, the routine hangs up on the external port. In decision block 1612, if this process has no holes or the estimated diameter is greater than one (i.e., in the large regime), then the routine continues at block 1613, else the routine continues at block 1616. In blocks 1613-1615, the routine loops forwarding a request for an edge through which to connect to the calling process to the broadcast channel. One request is forwarded for each pair of holes of the calling process that needs to be filled. In decision block 1613, if the number of holes of the calling process to be filled is greater than or equal to two, then the routine continues at block 1614, else the routine continues at block 1616. In block 1614, the routine invokes the forward connection edge search routine. The invoked routine is passed to an indication of the calling process and the random walk distance. In one embodiment, the distance is twice in the estimated diameter of the broadcast channel. In block 1614, the routine decrements the holes left to fill by two and loops to block 1613. In decision block 1616, if there is still a hole to fill, then the routine continues at block 1617, else the routine returns. In block 1617, the routine invokes the fill hole routine passing the identification of the calling process. The fill hole routine broadcasts a connection port search statement (i.e., `connection_port_search_stmt`) for a hole of a connected process through which the calling process can connect to the broadcast channel. The routine then returns.

FIG. 17 is a flow diagram illustrating the processing of the add neighbor routine in one embodiment. This routine adds the process calling on the external port as a neighbor to this process. In block 1701, the routine identifies the calling process on the external port. In block 1702, the routine sets a flag to indicate that the neighbor has not yet received the broadcast messages from this process. This flag is used to ensure that there are no gaps in the messages initially sent to the new neighbor. The external port becomes the internal port for this connection. In decision block 1703, if this process is in the seeking connection state, then this process is connecting to its first neighbor and the routine continues at block 1704, else the routine continues at block 1705. In block 1704, the routine sets the connection state of this process to partially connected. In block 1705, the routine adds the calling process to the list of neighbors of this process. In block 1706, the routine installs an internal dispatcher for the new neighbor. The internal dispatcher is invoked when a message is received from that new neighbor

through the internal port of that new neighbor. In decision block 1707, if this process buffered up messages while not fully connected, then the routine continues at block 1708, else the routine continues at block 1709. In one embodiment, a process that is partially connected may buffer the messages that it receives through an internal connection so that it can send these messages as it connects to new neighbors. In block 1708, the routine sends the buffered messages to the new neighbor through the internal port. In decision block 1709, if the number of holes of this process equals the expected number of holes, then this process is fully connected and the routine continues at block 1710, else the routine continues at block 1711. In block 1710, the routine invokes the achieve connected routine to indicate that this process is fully connected. In decision block 1711, if the number of holes for this process is zero, then the routine continues at block 1712, else the routine returns. In block 1712, the routine deletes any pending edges and then returns. A pending edge is an edge that has been proposed to this process for edge pinning, which in this case is no longer needed.

FIG. 18 is a flow diagram illustrating the processing of the forward connection edge search routine in one embodiment. This routine is responsible for passing along a request to connect a requesting process to a randomly selected neighbor of this process through the internal port of the selected neighbor, that is part of the random walk. In decision block 1801, if the forwarding distance remaining is greater than zero, then the routine continues at block 1804, else the routine continues at block 1802. In decision block 1802, if the number of neighbors of this process is greater than one, then the routine continues at block 1804, else this broadcast channel is in the small regime and the routine continues at block 1803. In decision block 1803, if the requesting process is a neighbor of this process, then the routine returns, else the routine continues at block 1804. In blocks 1804-1807, the routine loops attempting to send a connection edge search call internal message (i.e., connection_edge_search_call) to a randomly selected neighbor. In block 1804, the routine randomly selects a neighbor of this process. In decision block 1805, if all the neighbors of this process have already been selected, then the routine cannot forward the message and the routine returns, else the routine continues at block 1806. In block 1806, the routine sends a connection edge search call internal message to the selected neighbor. In decision block 1807, if the sending of the message is successful, then the routine continues at block 1808, else the routine loops to block 1804 to select the next neighbor. When the sending of an internal message is unsuccessful, then the neighbor may have disconnected from the broadcast channel in an unplanned manner. Whenever such a situation is detected by the broadcaster component, it attempts to find another neighbor by invoking the fill holes routine to fill a single hole or the forward connecting edge search routine to fill two holes. In block 1808, the routine notes that the recently sent connection edge search call has not yet been acknowledged and indicates that the edge to this neighbor is reserved if the remaining forwarding distance is less than or equal to one. It is reserved because the selected neighbor may offer this edge to the requesting process for edge pinning. The routine then returns.

FIG. 19 is a flow diagram illustrating the processing of the handle edge proposal call routine. This routine is invoked when a message is received from a proposing process that proposes to connect an edge between the proposing process and one of its neighbors to this process for edge pinning. In decision block 1901, if the number of holes of this process

minus the number of pending edges is greater than or equal to one, then this process still has holes to be filled and the routine continues at block 1902, else the routine continues at block 1911. In decision block 1902, if the proposing process or its neighbor is a neighbor of this process, then the routine continues at block 1911, else the routine continues at block 1903. In block 1903, the routine indicates that the edge is pending between this process and the proposing process. In decision block 1904, if a proposed neighbor is already pending as a proposed neighbor, then the routine continues at block 1911, else the routine continues at block 1907. In block 1907, the routine sends an edge proposal response as an external message to the proposing process (i.e., edge_proposal_resp) indicating that the proposed edge is accepted. In decision block 1908, if the sending of the message was successful, then the routine continues at block 1909, else the routine returns. In block 1909, the routine adds the edge as a pending edge. In block 1910, the routine invokes the add neighbor routine to add the proposing process on the external port as a neighbor. The routine then returns. In block 1911, the routine sends an external message (i.e., edge_proposal_resp) indicating that this proposed edge is not accepted. In decision block 1912, if the number of holes is odd, then the routine continues at block 1913, else the routine returns. In block 1913, the routine invokes the fill hole routine and then returns.

FIG. 20 is a flow diagram illustrating the processing of the handle port connection call routine in one embodiment. This routine is invoked when an external message is received then indicates that the sending process wants to connect to one hole of this process. In decision block 2001, if the number of holes of this process is greater than zero, then the routine continues at block 2002, else the routine continues at block 2003. In decision block 2002, if the sending process is not a neighbor, then the routine continues at block 2004, else the routine continues to block 2003. In block 2003, the routine sends a port connection response external message (i.e., port_connection_resp) to the sending process that indicates that it is not okay to connect to this process. The routine then returns. In block 2004, the routine sends a port connection response external message to the sending process that indicates that is okay to connect to this process. In decision block 2005, if the sending of the message was successful, then the routine continues at block 2006, else the routine continues at block 2007. In block 2006, the routine invokes the add neighbor routine to add the sending process as a neighbor of this process and then returns. In block 2007, the routine hangs up the external connection. In block 2008, the routine invokes the connect request routine to request that a process connect to one of the holes of this process. The routine then returns.

FIG. 21 is a flow diagram illustrating the processing of the fill hole routine in one embodiment. This routine is passed an indication of the requesting process. If this process is requesting to fill a hole, then this routine sends an internal message to other processes. If another process is requesting to fill a hole, then this routine invokes the routine to handle a connection port search request. In block 2101, the routine initializes a connection port search statement internal message (i.e., connection_port_search_stmt). In decision block 2102, if this process is the requesting process, then the routine continues at block 2103, else the routine continues at block 2104. In block 2103, the routine distributes the message to the neighbors of this process through the internal ports and then returns. In block 2104, the routine invokes the handle connection port search routine and then returns.

FIG. 22 is a flow diagram illustrating the processing of the internal dispatcher routine in one embodiment. This routine

is passed an indication of the neighbor who sent the internal message. In block 2201, the routine receives the internal message. This routine identifies the message type and invokes the appropriate routine to handle the message. In block 2202, the routine assesses whether to change the estimated diameter of the broadcast channel based on the information in the received message. In decision block 2203, if this process is the originating process of the message or the message has already been received (i.e., a duplicate), then the routine ignores the message and continues at block 2208, else the routine continues at block 2203A. In decision block 2203A, if the process is partially connected, then the routine continues at block 2203B, else the routine continues at block 2204. In block 2203B, the routine adds the message to the pending connection buffer and continues at block 2204. In decision blocks 2204-2207, the routine decodes the message type and invokes the appropriate routine to handle the message. For example, in decision block 2204, if the type of the message is broadcast statement (i.e., broadcast_stmt), then the routine invokes the handle broadcast message routine in block 2205. After invoking the appropriate handling routine, the routine continues at block 2208. In decision block 2208, if the partially connected buffer is full, then the routine continues at block 2209, else the routine continues at block 2210. The broadcaster component collects all its internal messages in a buffer while partially connected so that it can forward the messages as it connects to new neighbors. If, however, that buffer becomes full, then the process assumes that it is now fully connected and that the expected number of connections was too high, because the broadcast channel is now in the small regime. In block 2209, the routine invokes the achieve connection routine and then continues in block 2210. In decision block 2210, if the application program message queue is empty, then the routine returns, else the routine continues at block 2212. In block 2212, the routine invokes the receive response routine passing the acquired message and then returns. The received response routine is a callback routine of the application program.

FIG. 23 is a flow diagram illustrating the processing of the handle broadcast message routine in one embodiment. This routine is passed an indication of the originating process, an indication of the neighbor who sent the broadcast message, and the broadcast message itself. In block 2301, the routine performs the out of order processing for this message. The broadcaster component queues messages from each originating process until it can send them in sequence number order to the application program. In block 2302, the routine invokes the distribute broadcast message routine to forward the message to the neighbors of this process. In decision block 2303, if a newly connected neighbor is waiting to receive messages, then the routine continues at block 2304, else the routine returns. In block 2304, the routine sends the messages in the correct order if possible for each originating process and then returns.

FIG. 24 is a flow diagram illustrating the processing of the distribute broadcast message routine in one embodiment. This routine sends the broadcast message to each of the neighbors of this process, except for the neighbor who sent the message to this process. In block 2401, the routine selects the next neighbor other than the neighbor who sent the message. In decision block 2402, if all such neighbors have already been selected, then the routine returns. In block 2403, the routine sends the message to the selected neighbor and then loops to block 2401 to select the next neighbor.

FIG. 26 is a flow diagram illustrating the processing of the handle connection port search statement routine in one

embodiment. This routine is passed an indication of the neighbor that sent the message and the message itself. In block 2601, the routine invokes the distribute internal message which sends the message to each of its neighbors other than the sending neighbor. In decision block 2602, if the number of holes of this process is greater than zero, then the routine continues at block 2603, else the routine returns. In decision block 2603, if the requesting process is a neighbor, then the routine continues at block 2605, else the routine continues at block 2604. In block 2604, the routine invokes the court neighbor routine and then returns. The court neighbor routine connects this process to the requesting process if possible. In block 2605, if this process has one hole, then the neighbors with empty ports condition exists and the routine continues at block 2606, else the routine returns. In block 2606, the routine generates a condition check message (i.e., condition_check) that includes a list of this process' neighbors. In block 2607, the routine sends the message to the requesting neighbor.

FIG. 27 is a flow diagram illustrating the processing of the court neighbor routine in one embodiment. This routine is passed an indication of the prospective neighbor for this process. If this process can connect to the prospective neighbor, then it sends a port connection call external message to the prospective neighbor and adds the prospective neighbor as a neighbor. In decision block 2701, if the prospective neighbor is already a neighbor, then the routine returns, else the routine continues at block 2702. In block 2702, the routine dials the prospective neighbor. In decision block 2703, if the number of holes of this process is greater than zero, then the routine continues at block 2704, else the routine continues at block 2706. In block 2704, the routine sends a port connection call external message (i.e., port_connection_call) to the prospective neighbor and receives its response (i.e., port_connection_resp). Assuming the response is successfully received, in block 2705, the routine adds the prospective neighbor as a neighbor of this process by invoking the add neighbor routine. In block 2706, the routine hangs up with the prospect and then returns.

FIG. 28 is a flow diagram illustrating the processing of the handle connection edge search call routine in one embodiment. This routine is passed a indication of the neighbor who sent the message and the message itself. This routine either forwards the message to a neighbor or proposes the edge between this process and the sending neighbor to the requesting process for edge pinning. In decision block 2801, if this process is not the requesting process or the number of holes of the requesting process is still greater than or equal to two, then the routine continues at block 2802, else the routine continues at block 2813. In decision block 2802, if the forwarding distance is greater than zero, then the random walk is not complete and the routine continues at block 2803, else the routine continues at block 2804. In block 2803, the routine invokes the forward connection edge search routine passing the identification of the requesting process and the decremented forwarding distance. The routine then continues at block 2815. In decision block 2804, if the requesting process is a neighbor or the edge between this process and the sending neighbor is reserved because it has already been offered to a process, then the routine continues at block 2805, else the routine continues at block 2806. In block 2805, the routine invokes the forward connection edge search routine passing an indication of the requesting party and a toggle indicator that alternatively indicates to continue the random walk for one or two more computers. The routine then continues at block 2815. In block 2806, the routine dials the requesting process via the call-in port. In block 2807, the

routine sends an edge proposal call external message (i.e., `edge_proposal_call`) and receives the response (i.e., `edge_proposal_resp`). Assuming that the response is successfully received, the routine continues at block 2808. In decision block 2808, if the response indicates that the edge is acceptable to the requesting process, then the routine continues at block 2809, else the routine continues at block 2812. In block 2809, the routine reserves the edge between this process and the sending neighbor. In block 2810, the routine adds the requesting process as a neighbor by invoking the add neighbor routine. In block 2811, the routine removes the sending neighbor as a neighbor. In block 2812, the routine hangs up the external port and continues at block 2815. In decision block 2813, if this process is the requesting process and the number of holes of this process equals one, then the routine continues at block 2814, else the routine continues at block 2815. In block 2814, the routine invokes the fill hole routine. In block 2815, the routine sends an connection edge search response message (i.e., `connection_edge_search_response`) to the sending neighbor indicating acknowledgement and then returns. The graphs are sensitive to parity. That is, all possible paths starting from a node and ending at that node will have an even length unless the graph has a cycle whose length is odd. The broadcaster component uses a toggle indicator to vary the random walk distance between even and odd distances.

FIG. 29 is a flow diagram illustrating the processing of the handle connection edge search response routine in one embodiment. This routine is passed as indication of the requesting process, the sending neighbor, and the message. In block 2901, the routine notes that the connection edge search response (i.e., `connection_edge_search_resp`) has been received and if the forwarding distance is less than or equal to one unreserves the edge between this process and the sending neighbor. In decision block 2902, if the requesting process indicates that the edge is acceptable as indicated in the message, then the routine continues at block 2903, else the routine returns. In block 2903, the routine reserves the edge between this process and the sending neighbor. In block 2904, the routine removes the sending neighbor as a neighbor. In block 2905, the routine invokes the court neighbor routine to connect to the requesting process. In decision block 2906, if the invoked routine was unsuccessful, then the routine continues at block 2907, else the routine returns. In decision block 2907, if the number of holes of this process is greater than zero, then the routine continues at block 2908, else the routine returns. In block 2908, the routine invokes the fill hole routine and then returns.

FIG. 30 is a flow diagram illustrating the processing of the broadcast routine in one embodiment. This routine is invoked by the application program to broadcast a message on the broadcast channel. This routine is passed the message to be broadcast. In decision block 3001, if this process has at least one neighbor, then the routine continues at block 3002, else the routine returns since it is the only process connected to be broadcast channel. In block 3002, the routine generates an internal message of the broadcast statement type (i.e., `broadcast_stmt`). In block 3003, the routine sets the sequence number of the message. In block 3004, the routine invokes the distribute internal message routine to broadcast the message on the broadcast channel. The routine returns.

FIG. 31 is a flow diagram illustrating the processing of the acquire message routine in one embodiment. The acquire message routine may be invoked by the application program or by a callback routine provided by the application pro-

gram. This routine returns a message. In block 3101, the routine pops the message from the message queue of the broadcast channel. In decision block 3102, if a message was retrieved, then the routine returns an indication of success, else the routine returns indication of failure.

FIGS. 32-34 are flow diagrams illustrating the processing of messages associated with the neighbors with empty ports condition. FIG. 32 is a flow diagram illustrating processing of the handle condition check message in one embodiment. This message is sent by a neighbor process that has one hole and has received a request to connect to a hole of this process. In decision block 3201, if the number of holes of this process is equal to one, then the routine continues at block 3202, else the neighbors with empty ports condition does not exist any more and the routine returns. In decision block 3202, if the sending neighbor and this process have the same set of neighbors, the routine continues at block 3203, else the routine continues at block 3205. In block 3203, the routine initializes a condition double check message (i.e., `condition_double_check`) with the list of neighbors of this process. In block 3204, the routine sends the message internally to a neighbor other than sending neighbor. The routine then returns. In block 3205, the routine selects a neighbor of the sending process that is not also a neighbor of this process. In block 3206, the routine sends a condition repair message (i.e., `condition_repair_stmt`) externally to the selected process. In block 3207, the routine invokes the add neighbor routine to add the selected neighbor as a neighbor of this process and then returns.

FIG. 33 is a flow diagram illustrating processing of the handle condition repair statement routine in one embodiment. This routine removes an existing neighbor and connects to the process that sent the message. In decision block 3301, if this process has no holes, then the routine continues at block 3302, else the routine continues at block 3304. In block 3302, the routine selects a neighbor that is not involved in the neighbors with empty ports condition. In block 3303, the routine removes the selected neighbor as a neighbor of this process. Thus, this process that is executing the routine now has at least one hole. In block 3304, the routine invokes the add neighbor routine to add the process that sent the message as a neighbor of this process. The routine then returns.

FIG. 34 is a flow diagram illustrating the processing of the handle condition double check routine. This routine determines whether the neighbors with empty ports condition really is a problem or whether the broadcast channel is in the small regime. In decision block 3401, if this process has one hole, then the routine continues at block 3402, else the routine continues at block 3403. If this process does not have one hole, then the set of neighbors of this process is not the same as the set of neighbors of the sending process. In decision block 3402, if this process and the sending process have the same set of neighbors, then the broadcast channel is not in the small regime and the routine continues at block 3403, else the routine continues at block 3406. In decision block 3403, if this process has no holes, then the routine returns, else the routine continues at block 3404. In block 3404, the routine sets the estimated diameter for this process to one. In block 3405, the routine broadcasts a diameter reset internal message (i.e., `diameter_reset`) indicating that the estimated diameter is one and then returns. In block 3406, the routine creates a list of neighbors of this process. In block 3407, the routine sends the condition check message (i.e., `condition_check_stmt`) with the list of neighbors to the neighbor who sent the condition double check message and then returns.

From the above description, it will be appreciated that although specific embodiments of the technology have been

described, various modifications may be made without deviating from the spirit and scope of the invention. For example, the communications on the broadcast channel may be encrypted. Also, the channel instance or session identifier may be a very large number (e.g., 128 bits) to help prevent an unauthorized user to maliciously tap into a broadcast channel. The portal computer may also enforce security and not allow an unauthorized user to connect to the broadcast channel. Accordingly, the invention is not limited except by the claims.

What is claim is:

1. A non-routing table based computer network having a plurality of participants, each participant having connections to at least three neighbor participants, wherein an originating participant sends data to the other participants by sending the data through each of its connections to its neighbor participants, wherein each participant sends data that it receives from a neighbor participant to its other neighbor participants, wherein data is numbered sequentially so that data received out of order can be queued and rearranged, further wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.
2. The computer network of claim 1 wherein each participant is connected to 4 other participants.
3. The computer network of claim 1 wherein each participant is connected to an even number of other participants.
4. The computer network of claim 1 wherein all the participants are peers.
5. The computer network of claim 1 wherein the connections are peer-to-peer connections.
6. The computer network of claim 1 wherein the connections are TCP/IP connections.
7. The computer network of claim 1 wherein each participant is a process executing on a computer.
8. The computer network of claim 1 wherein a computer hosts more than one participant.
9. The computer network of claim 1 wherein each participant sends to each of its neighbors only one copy of the data.
10. A non-routing table based broadcast channel for participants, comprising:
 - a communications network that provides peer-to-peer communications between the participants connected to the broadcast channel; and
 - for each participant connected to the broadcast channel, an indication of four neighbor participants of that participant; and
 - a broadcast component that receives data from a neighbor participant using the communications network and that sends the received data to its other neighbor participants to effect the broadcasting of the data to each participant of the to broadcast channel, wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.

11. The broadcast channel of claim 10 wherein the broadcast component disregards received data that it has already sent to its neighbor participants.

12. The broadcast channel of claim 10 wherein a participant connects to the broadcast channel by contacting a participant already connected to the broadcast channel.

13. The broadcast channel of claim 10 wherein each participant is a computer process.

14. The broadcast channel of claim 10 wherein each participant is a computer thread.

15. The broadcast channel of claim 10 wherein each participant is a computer.

16. The broadcast channel of claim 10 wherein the communications network uses TCP/IP protocol.

17. The broadcast channel of claim 10 wherein the communications network is the Internet.

18. The broadcast channel of claim 10 wherein the participants are peers.

19. A non-routing table based computer-readable medium containing instructions for controlling communications of a participant of a broadcast channel within a network, by a method comprising:

- locating a portal computer;
- requesting the located portal computer to provide an indication of neighbor participants to which the participant can be connected;

- receiving the indications of the neighbor participants; and
- establishing a connection between the participant and each of the indicated neighbor participants, wherein a connection between the portal computer and the participant is not established, wherein a connection between the portal computer and the neighbor participants is not established, further wherein the network is m-regular and m-connected, where m is the number of neighbor participants of each participant, and further wherein the number of participants is at least two greater than m thus resulting in a non-complete graph.

20. The computer-readable medium of claim 19 wherein each participant is a computer process.

21. The computer-readable medium of claim 19 wherein the indicated participants are computer processes executing on different computer systems.

22. The computer-readable medium of claim 19 including:

- receiving data from a neighbor participant of the participant; and
- transmitting the received data to the other neighbor participants.

23. The computer-readable medium of claim 19 including:

- receiving a request to connect to another participant;
- disconnecting from a neighbor participant; and
- connecting to the other participant.

24. The computer-readable medium of claim 19 wherein the connections are established using the TCP/IP protocol.

* * * * *



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
 UNITED STATES PATENT AND TRADEMARK OFFICE
 WASHINGTON, D.C. 20231
 www.uspto.gov



Bib Data Sheet

SERIAL NUMBER 09/629,576	FILING DATE 07/31/2000	CLASS 370	GROUP ART UNIT 2731	ATTORNEY DOCKET NO. 030048001
APPLICANTS Fred B. Holt, Seattle, WA ; Virgil E. Bourassa, Bellevue, WA ;				
** CONTINUING DATA *****				
** FOREIGN APPLICATIONS *****				
IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** 09/22/2000				
Foreign Priority claimed 35 USC 119 (a-d) conditions met	<input type="checkbox"/> yes <input checked="" type="checkbox"/> no Allowance	STATE OR COUNTRY WA	SHEETS DRAWING 39	TOTAL CLAIMS 49
Verified and Acknowledged	Examiner's Signature <i>yw</i> Initials			INDEPENDENT CLAIMS 6
ADDRESS 25096				
TITLE Broadcasting network				
FILING FEE RECEIVED 1602	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:		<input type="checkbox"/> All Fees <input type="checkbox"/> 1.16 Fees (Filing) <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) <input type="checkbox"/> 1.18 Fees (Issue) <input type="checkbox"/> Other _____ <input type="checkbox"/> Credit	

REF ID: A66000

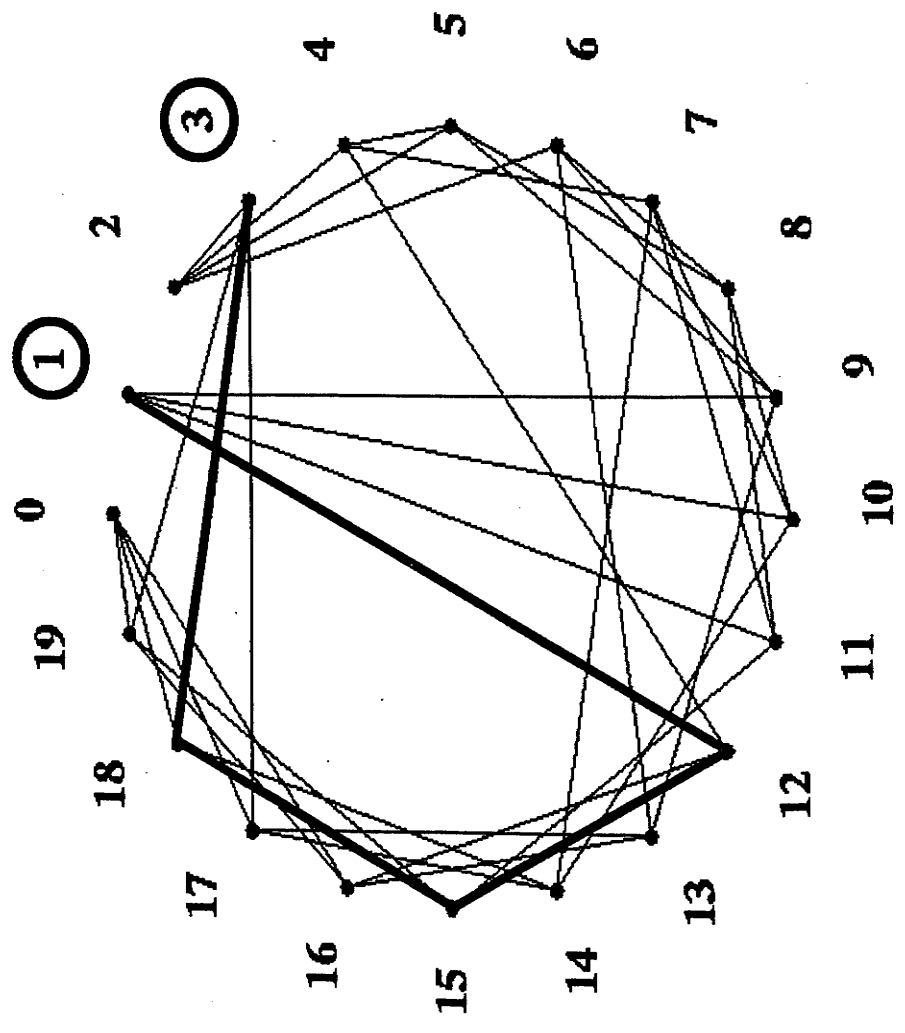


Fig 2

FIG. 3A

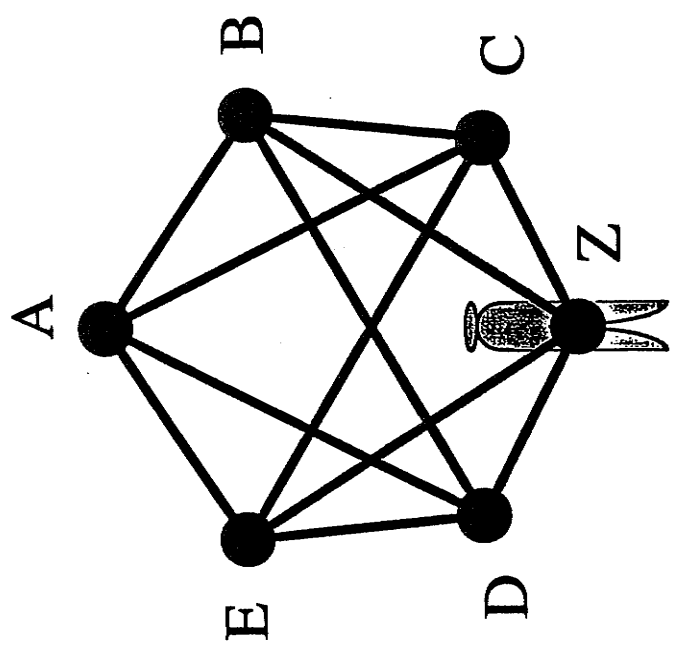
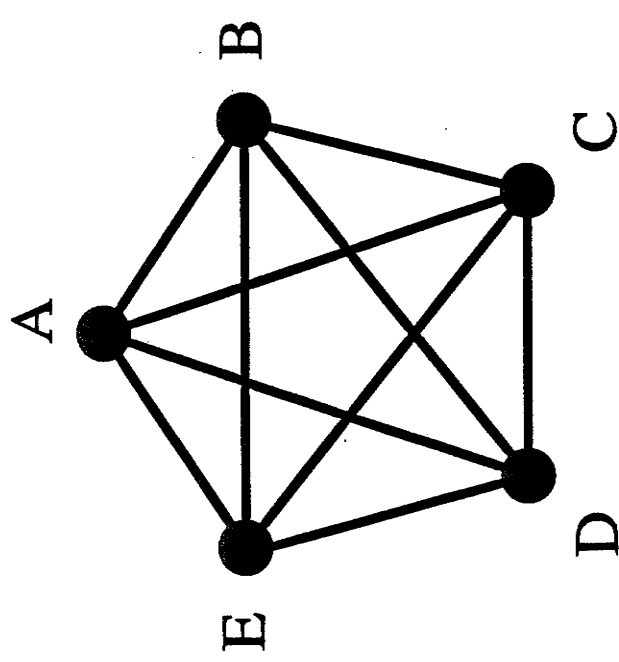


Fig 3A

Fig 3B

CONFIDENTIAL

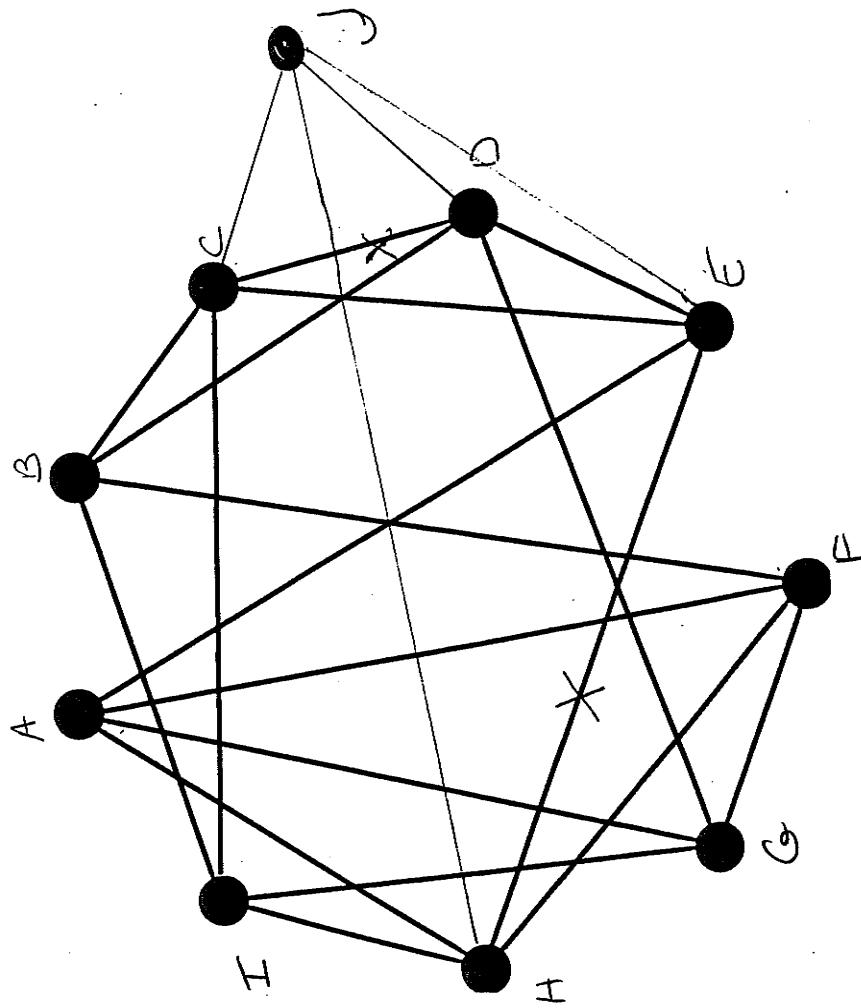


Fig 4A

0377

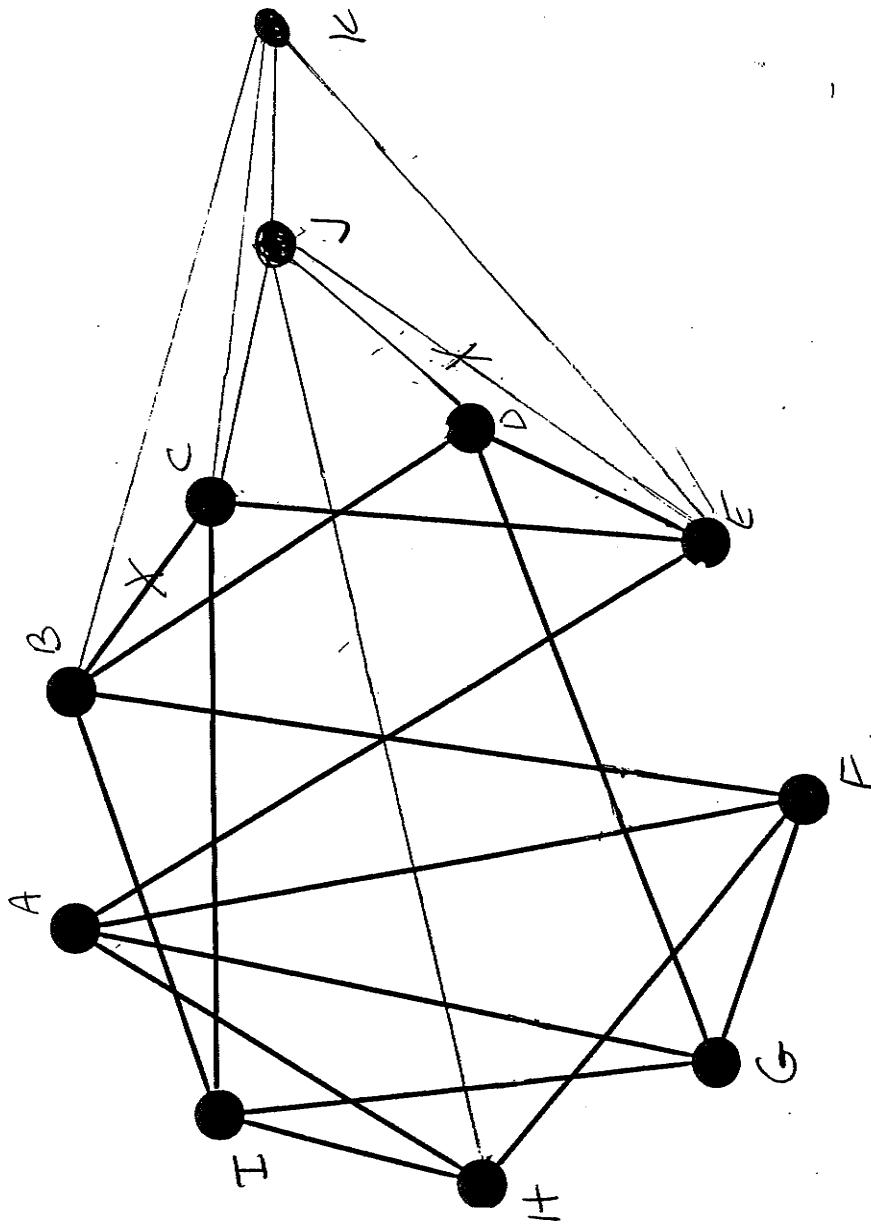


Fig 4B

0380 0380 0380

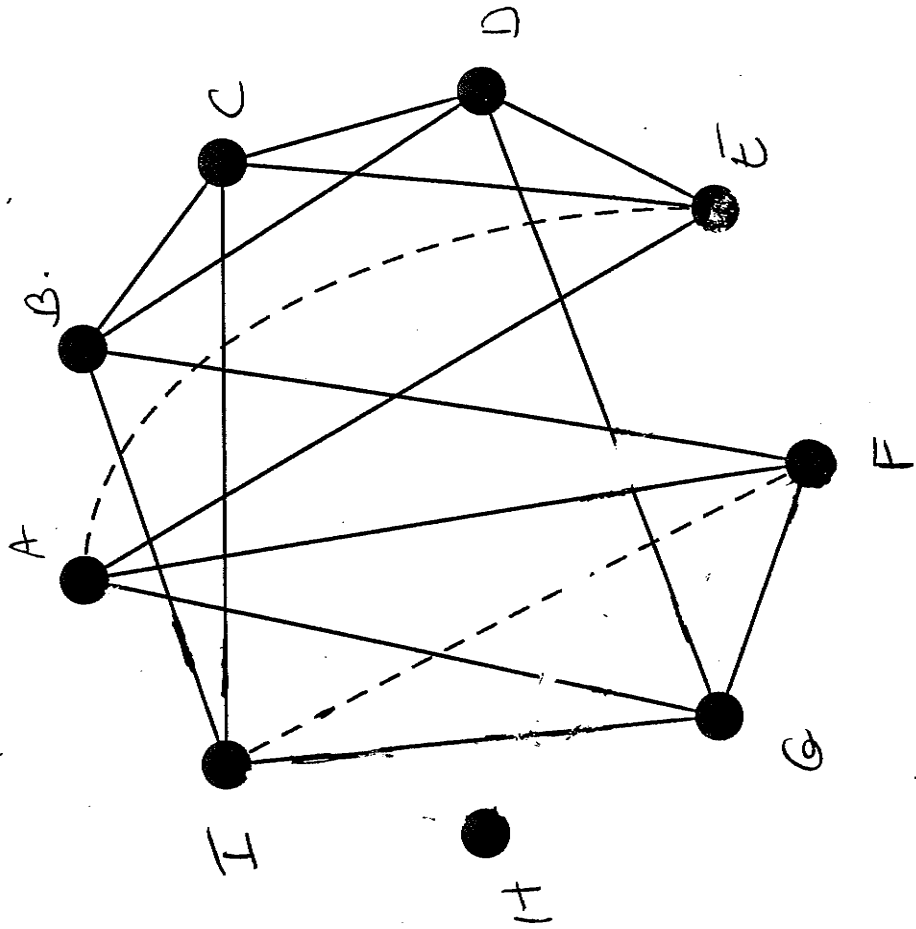


Fig 5B

CONFIDENTIAL

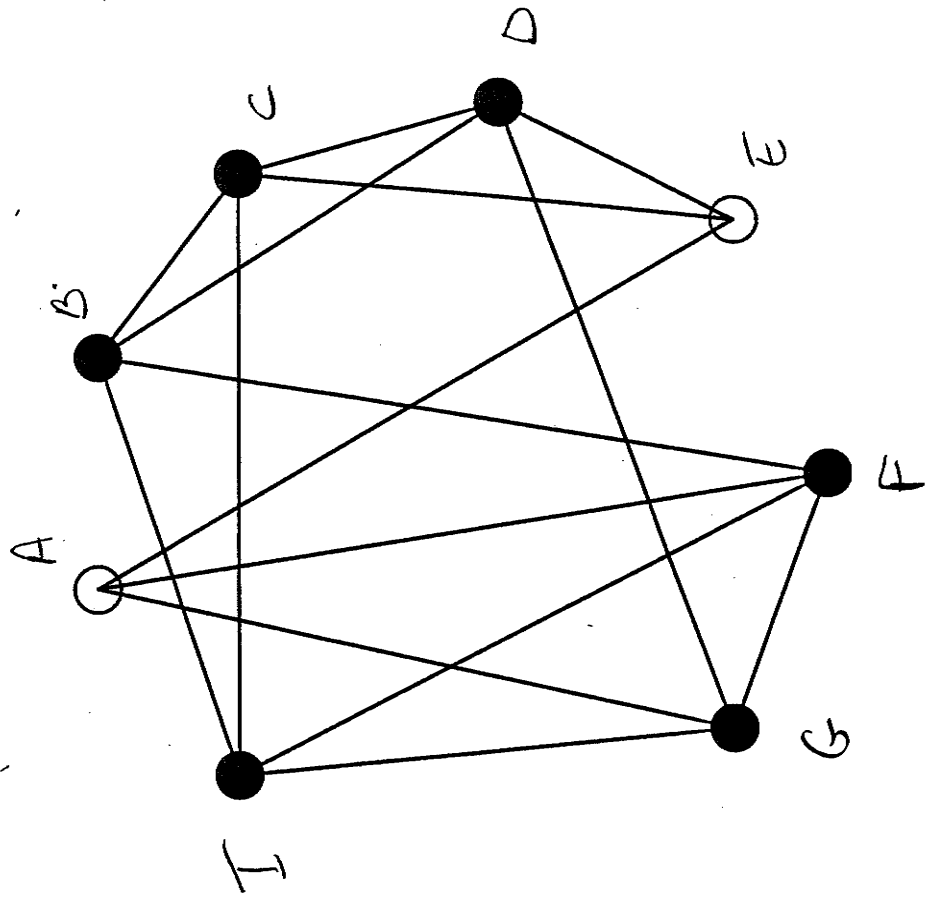


FIG 5C

00100000000000000000

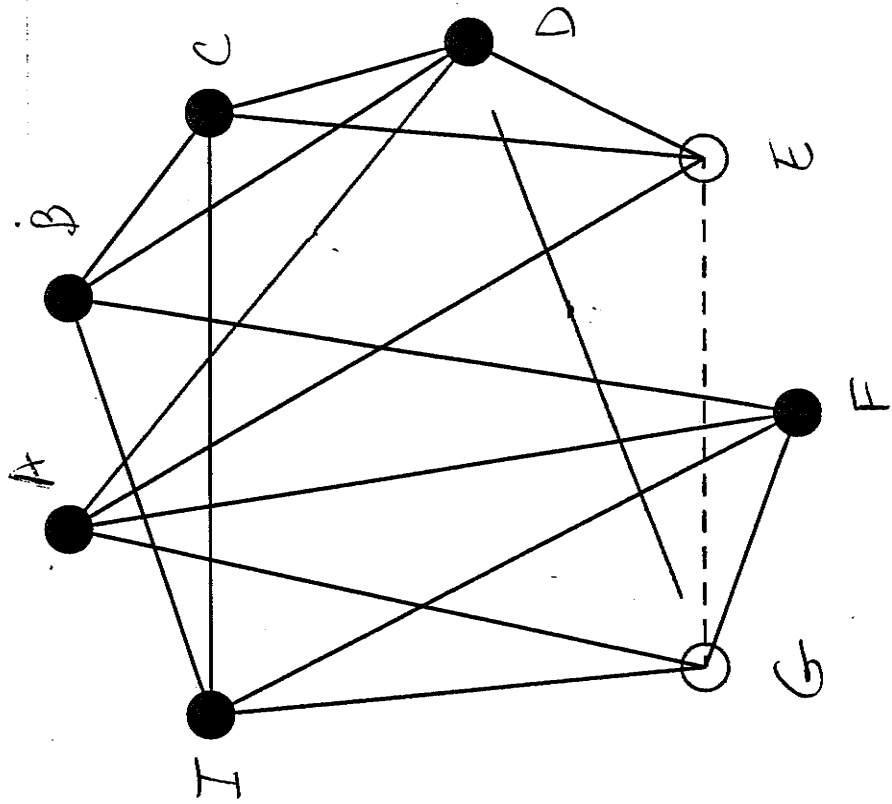


FIG 5D

00000000000000000000

00

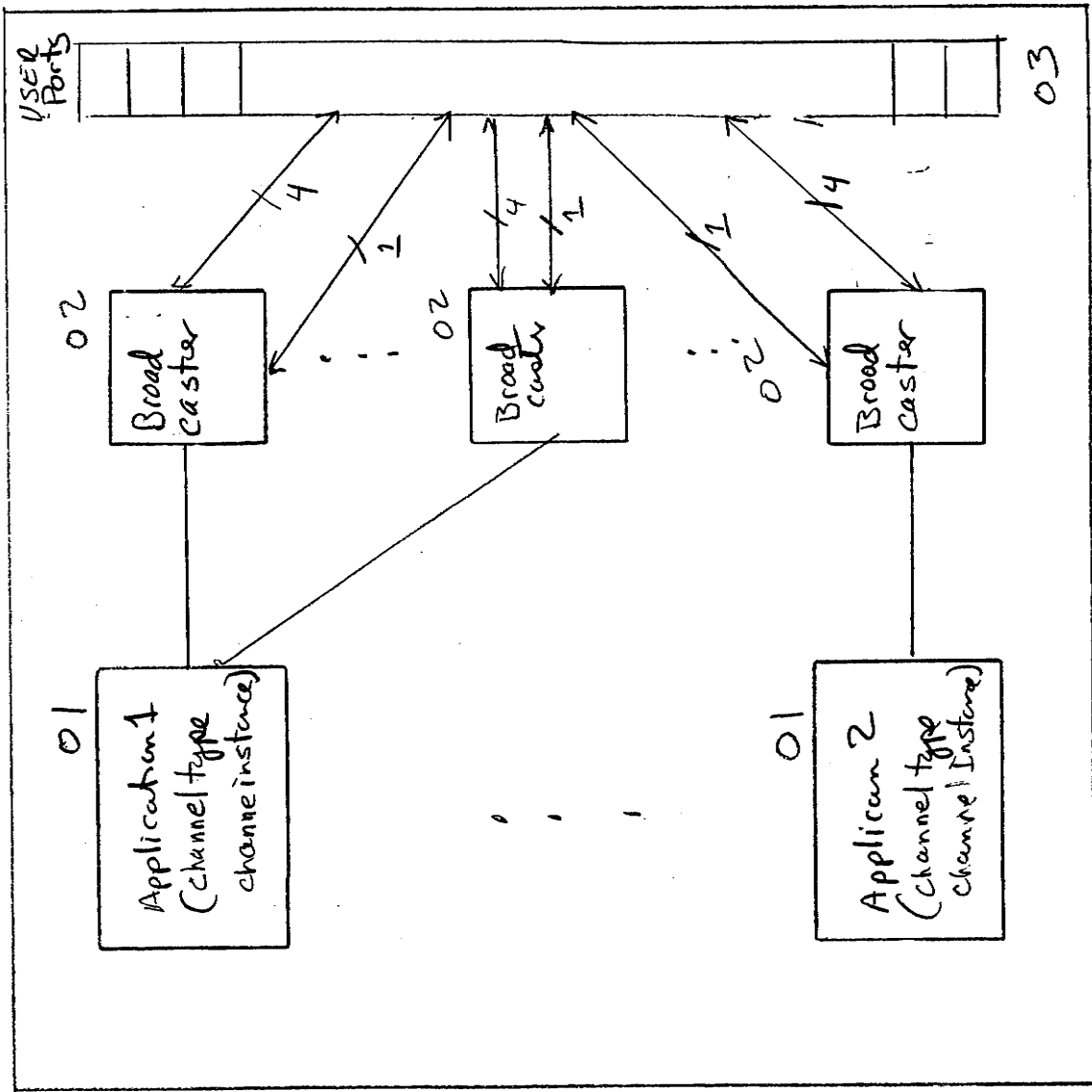


Fig 4

00

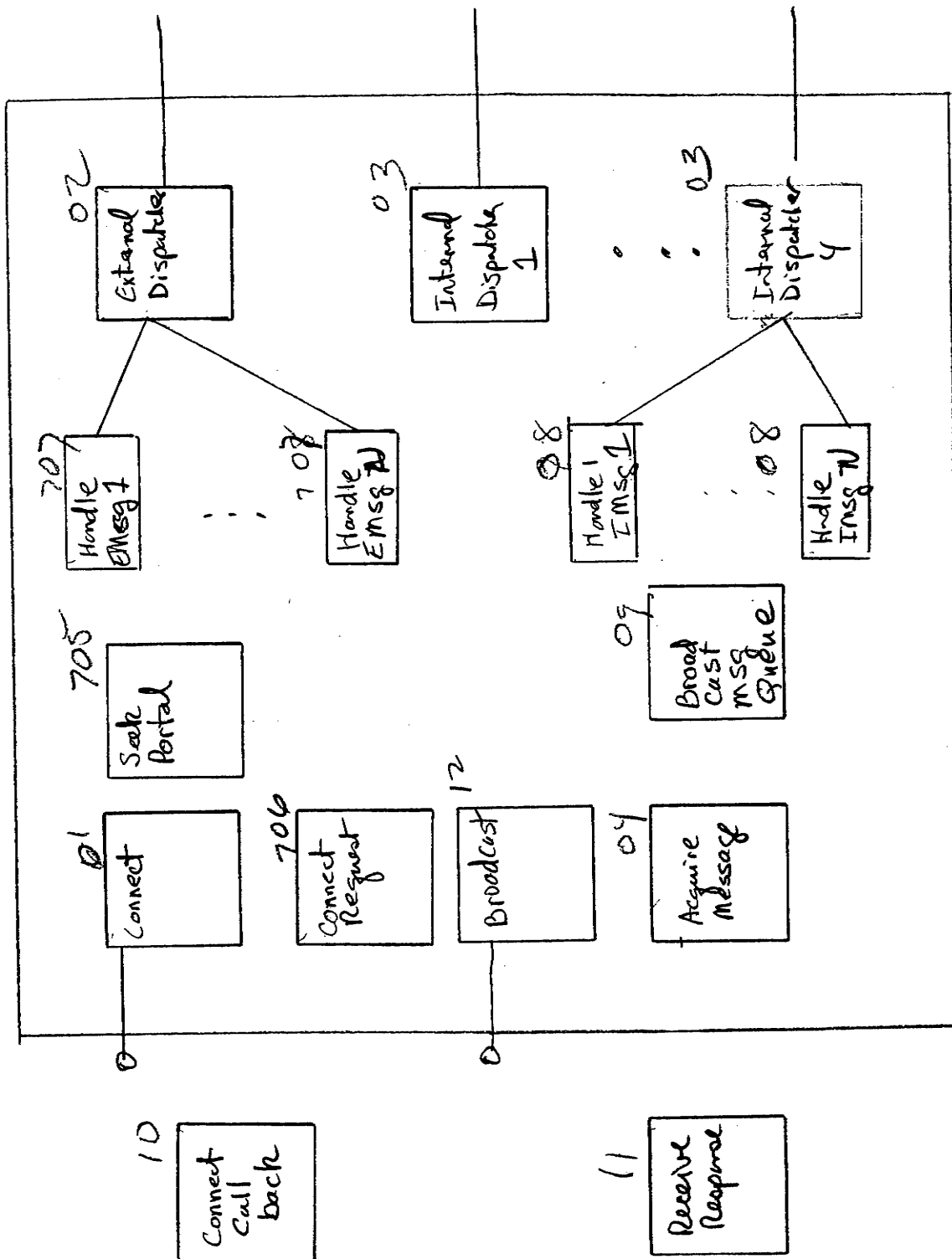


Figure 7

P4
Request

Channel type
channel Instance

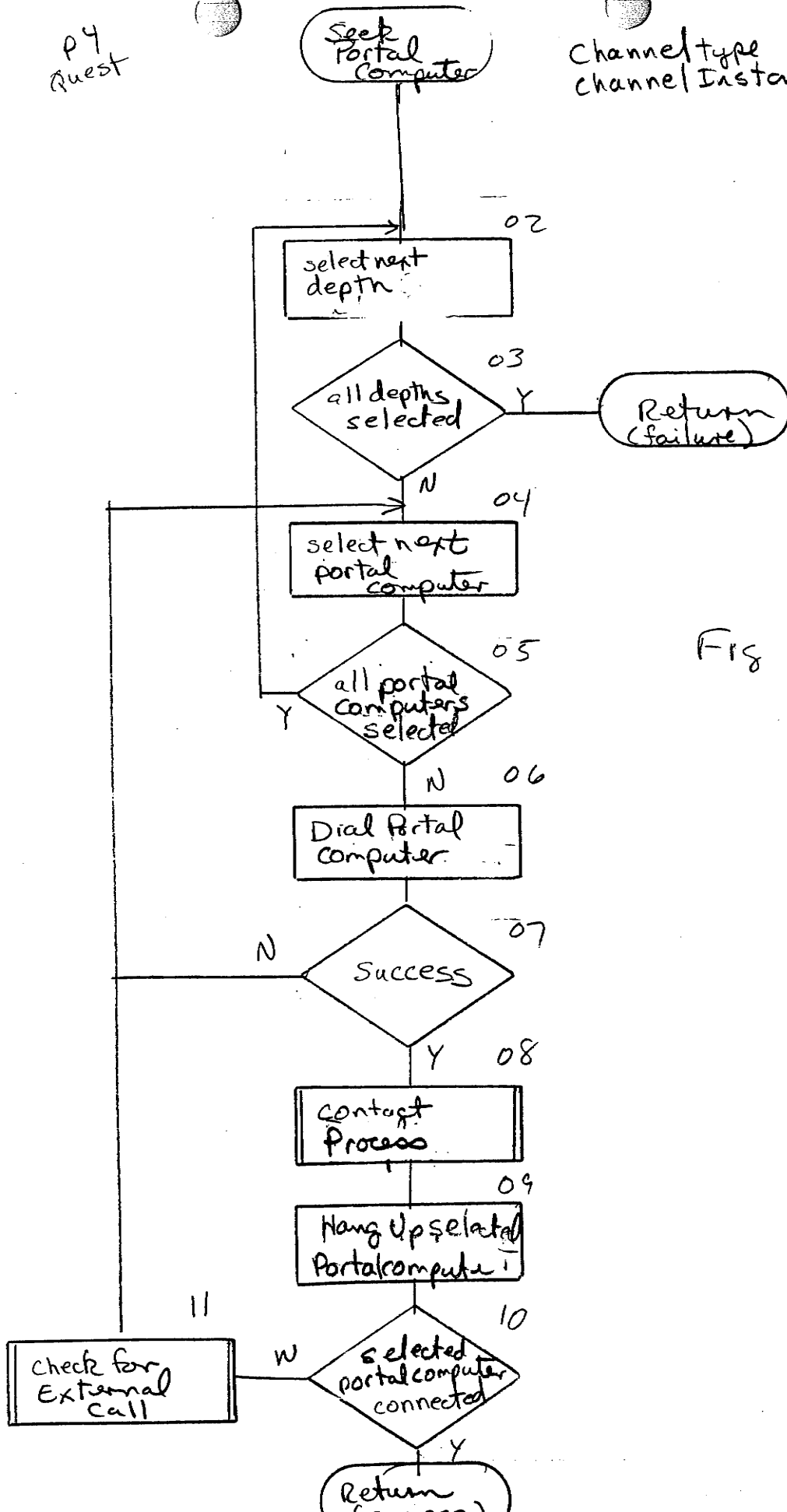


Fig 9

00120 21111111

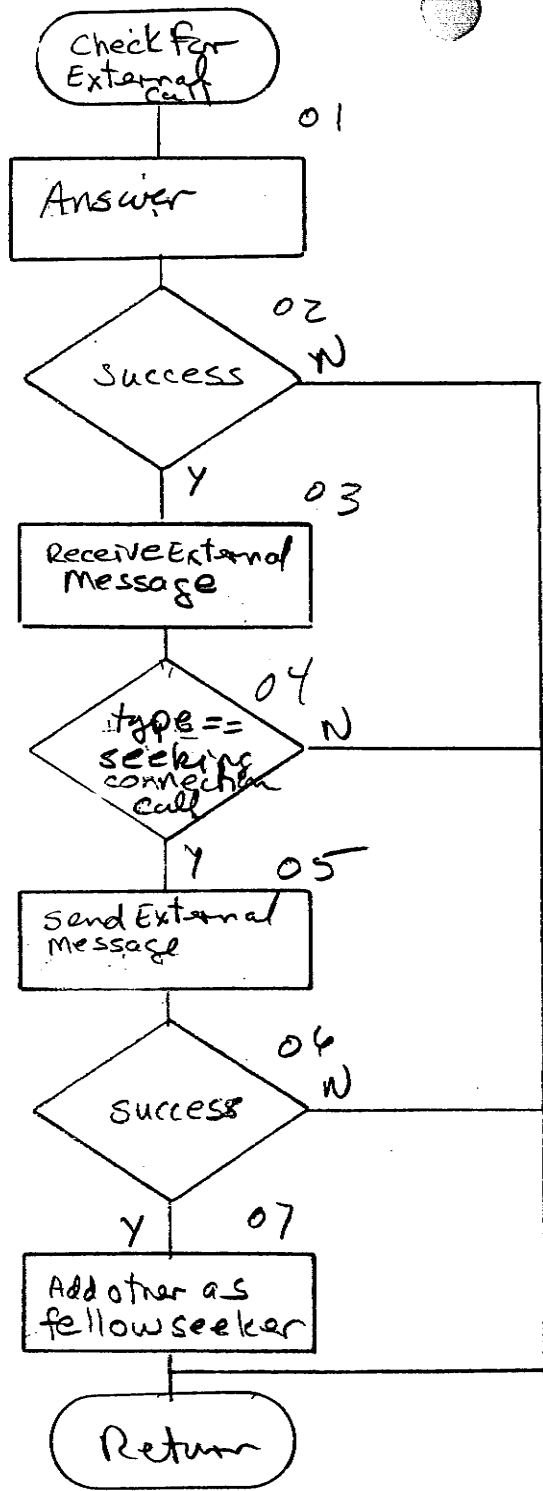


Fig 12

00000000000000000000

Achieve
Nirvana
22

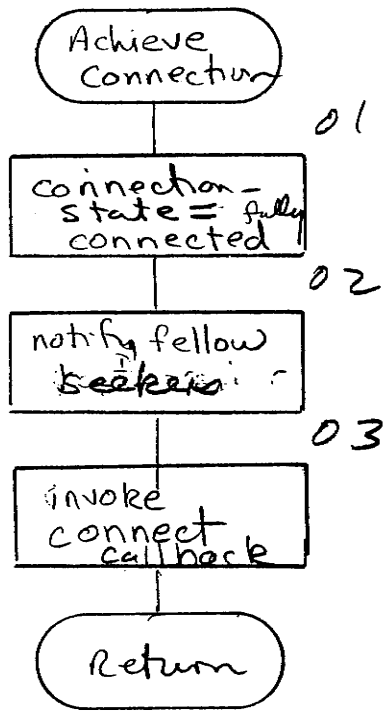
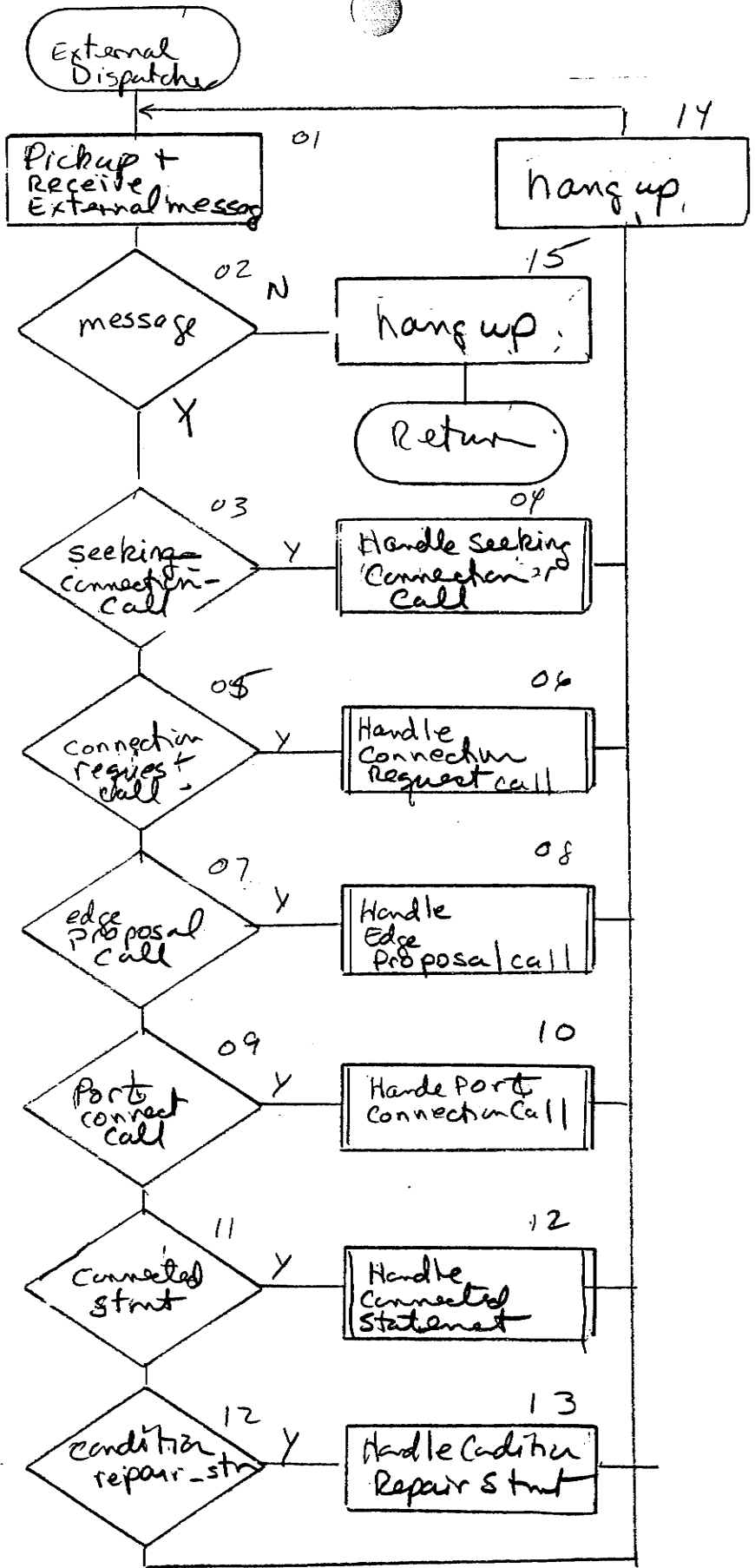


Fig 13

0391

P7

Fig 14

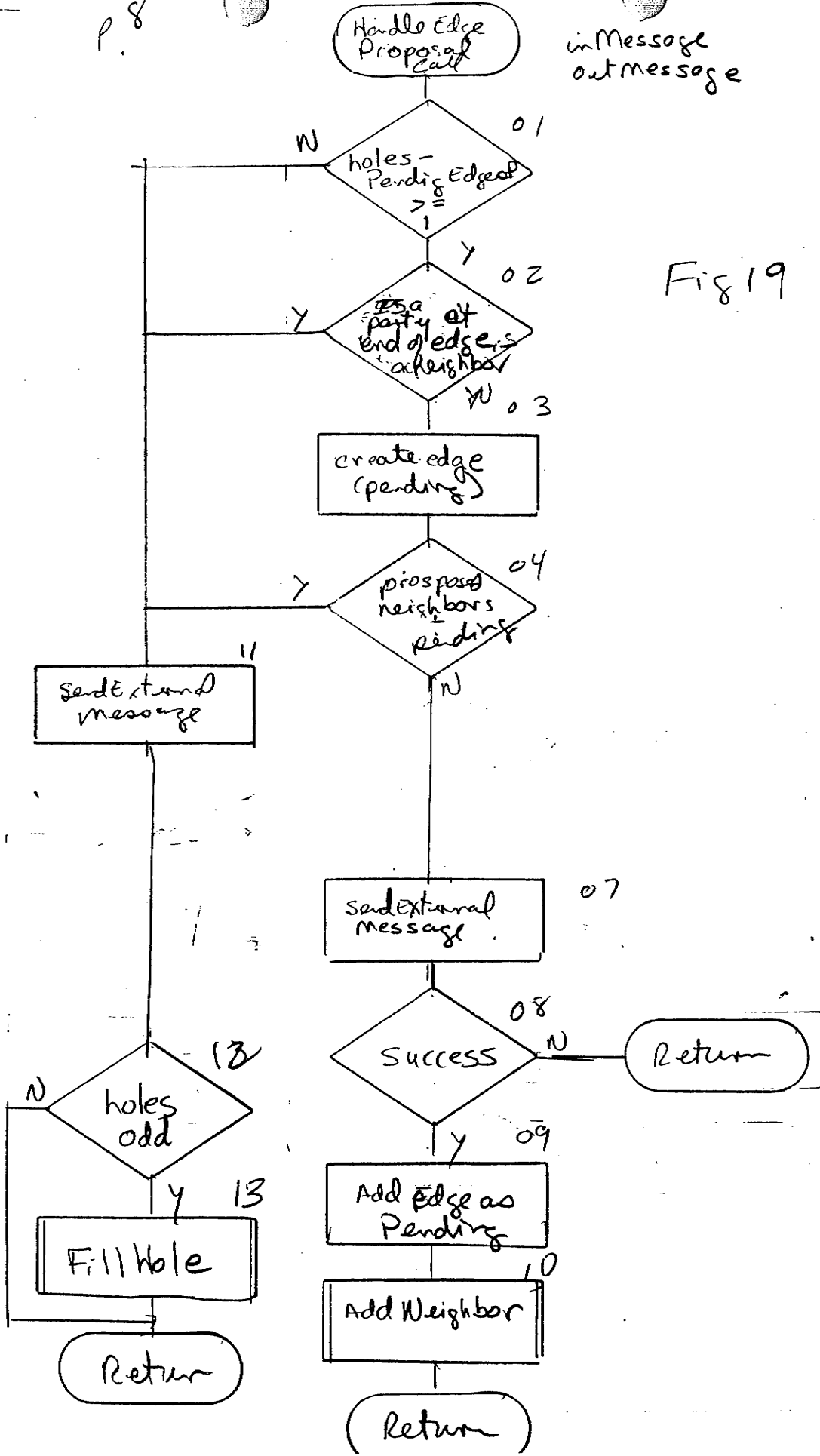


00110 9266666

p. 8

inMessage
outMessage

Fig 19



0011011000000000



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

<h1 style="margin: 0;">FEE TRANSMITTAL</h1> <h2 style="margin: 0;">for FY 2004</h2> <p style="font-size: small; margin: 5px 0;">Effective 10/01/2003. Patent fees are subject to annual revision.</p>	Complete if Known	
	Express Mail No.	EV336677851US
	Application Number	09/629,576
	Filing Date	July 31, 2000
	First Named Inventor	Fred B. Holt
	Examiner Name	Young N. Won
<input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27	Art Unit	2155
TOTAL AMOUNT OF PAYMENT (\$) 0	Attorney Docket No.	030048001US

RECEIVED

MAY 07 2004

Technology Center 2100

<p>METHOD OF PAYMENT (check all that apply)</p> <p><input type="checkbox"/> Check <input type="checkbox"/> Credit card <input type="checkbox"/> Money Order <input type="checkbox"/> Other <input type="checkbox"/> None</p> <p><input checked="" type="checkbox"/> Deposit Account: Deposit Account Number: <u>50-0665</u> Deposit Account Name: <u>Perkins Cole LLP</u></p> <p>The Commissioner is authorized to: (check all that apply)</p> <p><input type="checkbox"/> Charge fee(s) indicated below <input checked="" type="checkbox"/> Credit any overpayments <input checked="" type="checkbox"/> Charge any additional fee(s) during the pendency of this application <input type="checkbox"/> Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.</p> <p style="text-align: center;">FEE CALCULATION</p> <p>1. BASIC FILING FEE</p> <table border="1" style="width:100%; border-collapse: collapse; font-size: small;"> <thead> <tr> <th colspan="2">Large Entity</th> <th colspan="2">Small Entity</th> <th rowspan="2">Fee Description</th> <th rowspan="2">Fee Paid</th> </tr> <tr> <th>Code</th> <th>Fee (\$)</th> <th>Code</th> <th>Fee (\$)</th> </tr> </thead> <tbody> <tr><td>1001</td><td>770</td><td>2001</td><td>385</td><td>Utility filing fee</td><td></td></tr> <tr><td>1002</td><td>340</td><td>2002</td><td>170</td><td>Design filing fee</td><td></td></tr> <tr><td>1003</td><td>530</td><td>2003</td><td>265</td><td>Plant filing fee</td><td></td></tr> <tr><td>1004</td><td>770</td><td>2004</td><td>385</td><td>Reissue filing fee</td><td></td></tr> <tr><td>1205</td><td>160</td><td>2005</td><td>80</td><td>Provisional filing fee</td><td></td></tr> <tr> <td colspan="5" style="text-align: right;">SUBTOTAL (1)</td> <td style="text-align: center;">(\$)0</td> </tr> </tbody> </table> <p>2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE</p> <table border="1" style="width:100%; border-collapse: collapse; font-size: small;"> <thead> <tr> <th>Total Claims</th> <th>Extra Claims</th> <th>Fee from below</th> <th>Fee Paid</th> </tr> </thead> <tbody> <tr> <td>49</td> <td>49**</td> <td>X</td> <td>0</td> </tr> <tr> <td>6</td> <td>6**</td> <td>X</td> <td>0</td> </tr> </tbody> </table> <table border="1" style="width:100%; border-collapse: collapse; font-size: small;"> <thead> <tr> <th colspan="2">Large Entity</th> <th colspan="2">Small Entity</th> <th rowspan="2">Fee Description</th> <th rowspan="2">Fee Paid</th> </tr> <tr> <th>Code</th> <th>Fee (\$)</th> <th>Code</th> <th>Fee (\$)</th> </tr> </thead> <tbody> <tr><td>1202</td><td>18</td><td>2202</td><td>9</td><td>Claims in excess of 20</td><td></td></tr> <tr><td>1201</td><td>86</td><td>2201</td><td>43</td><td>Independent claims in excess of 3</td><td></td></tr> <tr><td>1203</td><td>290</td><td>2203</td><td>145</td><td>Multiple dependent claim, if not paid</td><td></td></tr> <tr><td>1204</td><td>86</td><td>2204</td><td>43</td><td>** Reissue independent claims over original patent</td><td></td></tr> <tr><td>1205</td><td>18</td><td>2205</td><td>9</td><td>** Reissue claims in excess of 20 and over original patent</td><td></td></tr> <tr> <td colspan="5" style="text-align: right;">SUBTOTAL (2)</td> <td style="text-align: center;">(\$)0</td> </tr> </tbody> </table> <p style="font-size: x-small;">**or number previously paid, if greater; For Reissues, see above</p>	Large Entity		Small Entity		Fee Description	Fee Paid	Code	Fee (\$)	Code	Fee (\$)	1001	770	2001	385	Utility filing fee		1002	340	2002	170	Design filing fee		1003	530	2003	265	Plant filing fee		1004	770	2004	385	Reissue filing fee		1205	160	2005	80	Provisional filing fee		SUBTOTAL (1)					(\$) 0	Total Claims	Extra Claims	Fee from below	Fee Paid	49	49**	X	0	6	6**	X	0	Large Entity		Small Entity		Fee Description	Fee Paid	Code	Fee (\$)	Code	Fee (\$)	1202	18	2202	9	Claims in excess of 20		1201	86	2201	43	Independent claims in excess of 3		1203	290	2203	145	Multiple dependent claim, if not paid		1204	86	2204	43	** Reissue independent claims over original patent		1205	18	2205	9	** Reissue claims in excess of 20 and over original patent		SUBTOTAL (2)					(\$) 0	<p>3. ADDITIONAL FEES</p> <table border="1" style="width:100%; border-collapse: collapse; font-size: small;"> <thead> <tr> <th colspan="2">Large Entity</th> <th colspan="2">Small Entity</th> <th rowspan="2">Fee Description</th> <th rowspan="2">Fee Paid</th> </tr> <tr> <th>Code</th> <th>Fee (\$)</th> <th>Code</th> <th>Fee (\$)</th> </tr> </thead> <tbody> <tr><td>1051</td><td>130</td><td>2051</td><td>65</td><td>Surcharge - late filing fee or oath</td><td></td></tr> <tr><td>1052</td><td>50</td><td>2052</td><td>25</td><td>Surcharge - late provisional filing fee or cover sheet</td><td></td></tr> <tr><td>1053</td><td>130</td><td>1053</td><td>130</td><td>Non-English Specification</td><td></td></tr> <tr><td>1812</td><td>2,520</td><td>1812</td><td>2,520</td><td>For filing a request for <i>ex parte</i> reexamination</td><td></td></tr> <tr><td>1804</td><td>920*</td><td>1804</td><td>920*</td><td>Requesting publication of SIR prior to Examiner action</td><td></td></tr> <tr><td>1805</td><td>1,840*</td><td>1805</td><td>1,840*</td><td>Requesting publication of SIR after Examiner action</td><td></td></tr> <tr><td>1251</td><td>110</td><td>2251</td><td>55</td><td>Extension for reply within first month</td><td></td></tr> <tr><td>1252</td><td>420</td><td>2252</td><td>210</td><td>Extension for reply within second month</td><td></td></tr> <tr><td>1253</td><td>950</td><td>2253</td><td>475</td><td>Extension for reply within third month</td><td></td></tr> <tr><td>1254</td><td>1,480</td><td>2254</td><td>740</td><td>Extension for reply within fourth month</td><td></td></tr> <tr><td>1255</td><td>2,010</td><td>2255</td><td>1,005</td><td>Extension for reply within fifth month</td><td></td></tr> <tr><td>1401</td><td>330</td><td>2401</td><td>165</td><td>Notice of Appeal</td><td></td></tr> <tr><td>1402</td><td>330</td><td>2402</td><td>165</td><td>Filing a brief in support of an appeal</td><td></td></tr> <tr><td>1403</td><td>290</td><td>2403</td><td>145</td><td>Request for oral hearing</td><td></td></tr> <tr><td>1451</td><td>1,510</td><td>1451</td><td>1,510</td><td>Petition to institute a public use proceeding</td><td></td></tr> <tr><td>1452</td><td>110</td><td>2452</td><td>55</td><td>Petition to revive - unavoidable</td><td></td></tr> <tr><td>1453</td><td>1,330</td><td>2453</td><td>665</td><td>Petition to revive - unintentional</td><td></td></tr> <tr><td>1501</td><td>1,330</td><td>2501</td><td>665</td><td>Utility issue fee (or reissue)</td><td></td></tr> <tr><td>1502</td><td>480</td><td>2502</td><td>240</td><td>Design issue fee</td><td></td></tr> <tr><td>1503</td><td>640</td><td>2503</td><td>320</td><td>Plant issue fee</td><td></td></tr> <tr><td>1460</td><td>130</td><td>1460</td><td>130</td><td>Petitions to the Commissioner</td><td></td></tr> <tr><td>1807</td><td>50</td><td>1807</td><td>50</td><td>Processing fee under 37 CFR 1.17(q)</td><td></td></tr> <tr><td>1806</td><td>180</td><td>1806</td><td>180</td><td>Submission of information Disclosure Stmt</td><td></td></tr> <tr><td>8021</td><td>40</td><td>8021</td><td>40</td><td>Recording each patent assignment per property (times number of properties)</td><td></td></tr> <tr><td>1809</td><td>770</td><td>2809</td><td>385</td><td>Filing a submission after final rejection (37 CFR 1.129(a))</td><td></td></tr> <tr><td>1810</td><td>770</td><td>2810</td><td>385</td><td>For each additional invention to be examined (37 CFR 1.129(b))</td><td></td></tr> <tr><td>1801</td><td>770</td><td>2801</td><td>385</td><td>Request for Continued Examination (RCE)</td><td></td></tr> <tr><td>1802</td><td>900</td><td>1802</td><td>900</td><td>Request for expedited examination of a design application</td><td></td></tr> </tbody> </table> <p>Other fee (specify) _____</p> <p>*Reduced by Basic Filing Fee Paid SUBTOTAL (3) (\$)0</p>	Large Entity		Small Entity		Fee Description	Fee Paid	Code	Fee (\$)	Code	Fee (\$)	1051	130	2051	65	Surcharge - late filing fee or oath		1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet		1053	130	1053	130	Non-English Specification		1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination		1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action		1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action		1251	110	2251	55	Extension for reply within first month		1252	420	2252	210	Extension for reply within second month		1253	950	2253	475	Extension for reply within third month		1254	1,480	2254	740	Extension for reply within fourth month		1255	2,010	2255	1,005	Extension for reply within fifth month		1401	330	2401	165	Notice of Appeal		1402	330	2402	165	Filing a brief in support of an appeal		1403	290	2403	145	Request for oral hearing		1451	1,510	1451	1,510	Petition to institute a public use proceeding		1452	110	2452	55	Petition to revive - unavoidable		1453	1,330	2453	665	Petition to revive - unintentional		1501	1,330	2501	665	Utility issue fee (or reissue)		1502	480	2502	240	Design issue fee		1503	640	2503	320	Plant issue fee		1460	130	1460	130	Petitions to the Commissioner		1807	50	1807	50	Processing fee under 37 CFR 1.17(q)		1806	180	1806	180	Submission of information Disclosure Stmt		8021	40	8021	40	Recording each patent assignment per property (times number of properties)		1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))		1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))		1801	770	2801	385	Request for Continued Examination (RCE)		1802	900	1802	900	Request for expedited examination of a design application	
Large Entity		Small Entity		Fee Description			Fee Paid																																																																																																																																																																																																																																																																																				
Code	Fee (\$)	Code	Fee (\$)																																																																																																																																																																																																																																																																																								
1001	770	2001	385	Utility filing fee																																																																																																																																																																																																																																																																																							
1002	340	2002	170	Design filing fee																																																																																																																																																																																																																																																																																							
1003	530	2003	265	Plant filing fee																																																																																																																																																																																																																																																																																							
1004	770	2004	385	Reissue filing fee																																																																																																																																																																																																																																																																																							
1205	160	2005	80	Provisional filing fee																																																																																																																																																																																																																																																																																							
SUBTOTAL (1)					(\$) 0																																																																																																																																																																																																																																																																																						
Total Claims	Extra Claims	Fee from below	Fee Paid																																																																																																																																																																																																																																																																																								
49	49**	X	0																																																																																																																																																																																																																																																																																								
6	6**	X	0																																																																																																																																																																																																																																																																																								
Large Entity		Small Entity		Fee Description	Fee Paid																																																																																																																																																																																																																																																																																						
Code	Fee (\$)	Code	Fee (\$)																																																																																																																																																																																																																																																																																								
1202	18	2202	9	Claims in excess of 20																																																																																																																																																																																																																																																																																							
1201	86	2201	43	Independent claims in excess of 3																																																																																																																																																																																																																																																																																							
1203	290	2203	145	Multiple dependent claim, if not paid																																																																																																																																																																																																																																																																																							
1204	86	2204	43	** Reissue independent claims over original patent																																																																																																																																																																																																																																																																																							
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent																																																																																																																																																																																																																																																																																							
SUBTOTAL (2)					(\$) 0																																																																																																																																																																																																																																																																																						
Large Entity		Small Entity		Fee Description	Fee Paid																																																																																																																																																																																																																																																																																						
Code	Fee (\$)	Code	Fee (\$)																																																																																																																																																																																																																																																																																								
1051	130	2051	65	Surcharge - late filing fee or oath																																																																																																																																																																																																																																																																																							
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet																																																																																																																																																																																																																																																																																							
1053	130	1053	130	Non-English Specification																																																																																																																																																																																																																																																																																							
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination																																																																																																																																																																																																																																																																																							
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action																																																																																																																																																																																																																																																																																							
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action																																																																																																																																																																																																																																																																																							
1251	110	2251	55	Extension for reply within first month																																																																																																																																																																																																																																																																																							
1252	420	2252	210	Extension for reply within second month																																																																																																																																																																																																																																																																																							
1253	950	2253	475	Extension for reply within third month																																																																																																																																																																																																																																																																																							
1254	1,480	2254	740	Extension for reply within fourth month																																																																																																																																																																																																																																																																																							
1255	2,010	2255	1,005	Extension for reply within fifth month																																																																																																																																																																																																																																																																																							
1401	330	2401	165	Notice of Appeal																																																																																																																																																																																																																																																																																							
1402	330	2402	165	Filing a brief in support of an appeal																																																																																																																																																																																																																																																																																							
1403	290	2403	145	Request for oral hearing																																																																																																																																																																																																																																																																																							
1451	1,510	1451	1,510	Petition to institute a public use proceeding																																																																																																																																																																																																																																																																																							
1452	110	2452	55	Petition to revive - unavoidable																																																																																																																																																																																																																																																																																							
1453	1,330	2453	665	Petition to revive - unintentional																																																																																																																																																																																																																																																																																							
1501	1,330	2501	665	Utility issue fee (or reissue)																																																																																																																																																																																																																																																																																							
1502	480	2502	240	Design issue fee																																																																																																																																																																																																																																																																																							
1503	640	2503	320	Plant issue fee																																																																																																																																																																																																																																																																																							
1460	130	1460	130	Petitions to the Commissioner																																																																																																																																																																																																																																																																																							
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)																																																																																																																																																																																																																																																																																							
1806	180	1806	180	Submission of information Disclosure Stmt																																																																																																																																																																																																																																																																																							
8021	40	8021	40	Recording each patent assignment per property (times number of properties)																																																																																																																																																																																																																																																																																							
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))																																																																																																																																																																																																																																																																																							
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))																																																																																																																																																																																																																																																																																							
1801	770	2801	385	Request for Continued Examination (RCE)																																																																																																																																																																																																																																																																																							
1802	900	1802	900	Request for expedited examination of a design application																																																																																																																																																																																																																																																																																							

SUBMITTED BY		(Complete (if applicable))	
Name (Print/Type)	Chun Ng	Registration No. (Attorney/Agent)	36-878
Signature		Telephone	206-359-8000
		Date	5/4/04

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-766-9199) and select option 2.

RELIABLE BROADCAST IN MOBILE WIRELESS NETWORKS¹

S. Alagar and S. Venkatesan
Department of Computer Science, University of Texas at Dallas
Richardson, TX 75083

J. R. Cleveland
C3 Systems Division, Electrospace Systems, Inc., A Chrysler Company
Richardson, TX 75081

ABSTRACT

This paper presents preliminary results of our research on wireless networking that supports reliable communications between nomadic hosts engaged in distributed computing and collaborative conferencing. Our network model consists of a set of low-power, radio frequency (RF) transceivers which move relative to each other across an irregular terrain subject to RF propagation impairments. The low transmitter power defines a radio coverage which limits the probability of intercept and the number of neighbors but optimizes frequency reuse. The combination of low power and propagation environment produces a network characterized by stochastic link failures. The rapidity of these failures and perturbations to the network topology defeats the use of routing policies based on maintaining routing tables or determining least cost paths. With these conditions as the background, our work addresses the need to provide reliable information exchange, mitigate bottlenecks, avoid excessive traffic, and offer scalable services without the benefit of static base station or fixed backbone support. Meeting such challenges demands a robust, flexible information transport system that delivers all required information for diverse operational scenarios. The approach emphasizes the importance of achieving guaranteed delivery across a network of limited size operating in a hostile environment rather than obtaining a high throughput per unit area, typical of commercial enterprises.

The basic premise of the protocol is that host mobility and terrain prevents *a priori* knowledge of any host location and optimum path. Message broadcasting, or flood routing, provides the means for reliable delivery of information in the presence of uncertain connectivity and node locations. Knowledge of the network results, instead, from a measure of transmitted and received message traffic. Central to the protocol is the provision for each mobile host to retain a *HISTORY* of messages broadcast to and received from its neighbor(s). A host which receives a message broadcasts an

acknowledgment to the sender, updates its local *HISTORY*, and then retransmits the message if it is not a duplicate message. Duplicated messages are discarded. If a sending host does not receive an acknowledgment from a neighbor within a certain time, it timeouts and resends the message. If a host does not receive an acknowledgment after several retries, it assumes that the link disconnection is not transient and stops sending the message. When a host detects a new neighbor, a handshake procedure results in the exchange of active messages not common to the respective *HISTORY* of each host. Once the handshake procedure terminates, the contents of the *HISTORY* for each host are identical. Thus, using handshake procedures, mobile hosts receive messages that they did not receive previously due to link disconnections. Idle hosts will periodically broadcast a sounding message to maintain their network presence.

1. **INTRODUCTION.** Winning the information war with complete and up-to-date intelligence is vital to the entire spectrum of possible operational requirements, whether engaged in war or corporate strategic planning. Military commanders engaged in rapid force projection, as well as public safety officers, medical staff, and corporate managers, demand accurate information regardless of location or situation. Each requires a clear and accurate picture of a changing situation to reach well-informed decisions and successful conclusion. Information must flow throughout the network toward the users at each level of the management hierarchy whether at the sustaining base or at the forward most part of the mission. Participating staff must have the capability to acquire or send accurate information that defines their space and situation. The information transport network must extend reliable voice, data, video, and imagery transmissions to nomadic users at any location. The availability of assured communications directly relates to mission success through computing, conferencing, and synchronized tasking while fixed or on-the-move. Invariably, this critical information is required when communications services normally provided by a reliable,

¹This research was supported in part by NSF under Grant No. CRR-9110177, by the Texas Advanced Technology Program under Grant No. 9741-036, and by a grant from Electrospace Systems, Inc., a Chrysler Company.

fixed infrastructure are unavailable or severely degraded.

The wireless networking of mobile (i.e., nomadic) subscribers is an emerging paradigm in the field of distributed command, control, and computing, with the potential to improve command and control responsiveness. In our context the phrase "mobile wireless network" means that not contain any static support stations. It supports the needs of subscribers to mobile and mobile satellite ground entry points. The network must provide reliable information, avoid bottlenecks, avoid excessive traffic, offer and, above all, adapt to dynamic topologies. The area should conform as closely as possible to which subscribers move, thereby offering detection and improving frequency implementation and operation are critical.

Nomadic wireless networks currently offer limited bandwidth and frequent changes. The challenge is to allow updates simultaneously, but only for those users in service. The major requirements include minimize updates, provide fault-tolerant service scalability, and minimize communication and computation overhead. Many of the algorithms that assume static hosts or well-defined point-to-point links cannot be directly used for mobile systems due to the changes in physical connectivity and limited bandwidth of the wireless links. This has spawned considerable research in mobile computing: designing communication protocols [1, 2, 3, 4], file system operations [5, 6], managing data efficiently [7, 8], and providing fault tolerance [9]. Most research on these topics is based on a model in which the mobile hosts are supported by static base stations such as cellular telephony or personal communications systems (PCS). A typical PCS topology takes the form of a single-hop network in which each host is within radio range of the base station or all other hosts. In this paper, we consider the problem of providing reliable broadcast in mobile wireless networks where single-hop and known topologies may not exist. Applications include disaster relief operations, highly mobile military or law enforcement operations, and rapid response contingency operations where it is not economical to place support stations.

2. WIRELESS NETWORK MODEL. The model of the mobile wireless network consists of several mobile hosts distributed over an irregular terrain (Figure 1). The mobile hosts use low-power transmitters and novel, efficient receivers [10] to communicate. Emerging technologies and products for PCS applications that operate in the ISM bands with a transmitter power of 1W [11] provide the basis for practical

implementation. In this network concept, the cell of a mobile host is the geographical area within which the mobile hosts can directly communicate with other mobile hosts. Note that the cell of a host does not remain fixed, but moves with each attached host. The nominal cell size (R) is determined by a path loss model that denotes the local average received signal power. A general path loss (PL) model is derived through measurement using the power law relationship $P_r = P_t (R_0/R)^n$. Based on both analysis and simulation, the microcell propagation model uses a characteristic distance R_0 . The power

8.2
 Description
 in
 Brief
 The Fig 8.5

$$P_r = P_t (R_0/R)^n + X_n \quad (1)$$

is at the characteristic distance R_0 is a Gaussian random variable that denotes the received power. Nelson and Nelson [11] and a slotted ALOHA network model show that the input occurs with a cell size less than an average of six nearest neighbors. The results produced in Figure 2. These results assume a random distribution of nodes across the terrain and a perfect capture condition. Perfect capture occurs when a node will always receive and detect the strongest of several simultaneous transmissions within its hearing range. The weaker signals appear as noise to the detection process. A non-capture condition occurs if simultaneous transmissions always result in collisions. The reduced power supports frequency reuse, as well as low probability of detection. Their analysis also shows that mobile hosts with sufficient

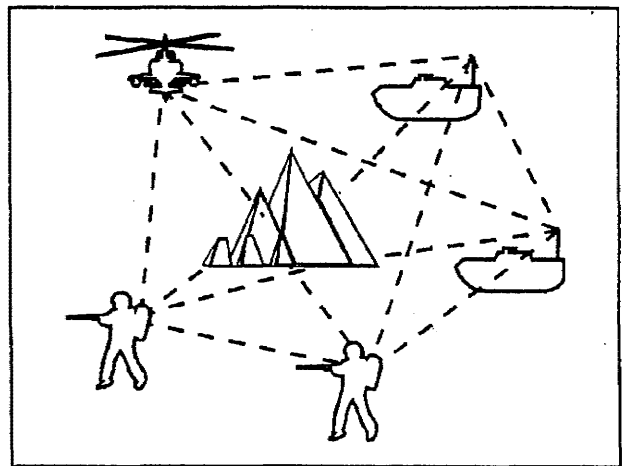


Figure 1. Model of a wireless computer network that experiences link failures due to range limitations and terrain impairments.

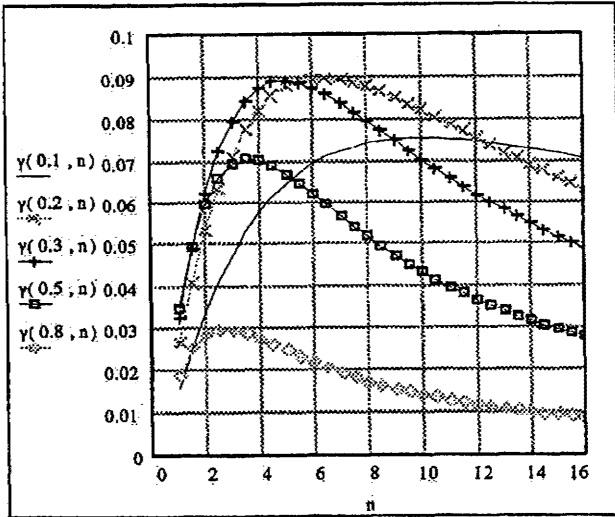


Figure 2. Normalized throughput for perfect capture with the slotted ALOHA protocol versus the average number of nearest neighbors for different probabilities that a node is busy.

transmitter power to reach all other hosts (i.e., a single-hop network) support a significantly lower throughput.

Detecting Neighbors. Neighbors are detected by a strategy common to a general class of survivable and adaptive network protocols that use sounding procedures [15, 16]. Two mobile hosts are *neighbors* if they can "hear" each other. Each host detects its neighbors by periodically broadcasting a probe

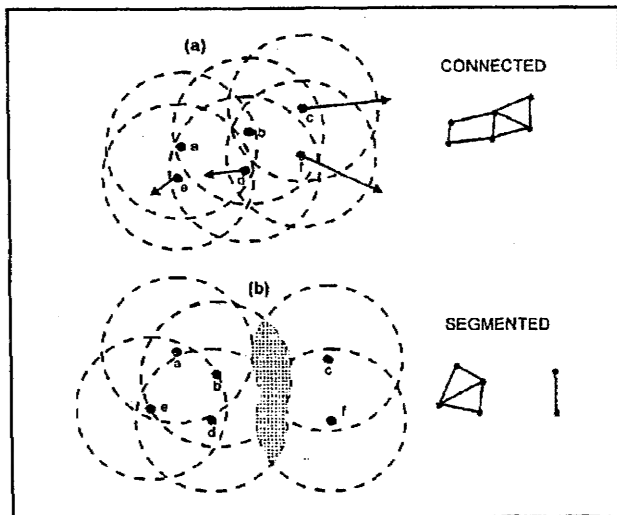


Figure 3. Examples of (a) a fully connected mobile wireless network and (b) a decomposed network due to mobility of hosts *c* and *f*. The shaded region indicates the common area within the range of hosts *c* and *f* and the rest of the network.

message. A host that hears a probe message sends an acknowledgment to the probing host. Every host maintains a list of neighbors and periodically updates the list based on acknowledgments received. When two hosts become neighbors, a wireless link is established between them, and they execute a *handshake* procedure. As part of the handshake procedure, they each update their list of neighbors.

Link Disconnections. The wireless link between two neighbors is unreliable due to RF propagation effects such as loss of line-of-sight (LOS), moving out of range, multipath fading, or inclement weather. There are two types of link disconnections: (1) transient and (2) permanent. In the transient case, a host is unable to communicate briefly with a neighbor due to: (a) the neighbor moving out of sight; (b) multipath fading; or (c) inclement weather. Multipath fading has a time dependence that varies from microseconds to seconds, depending on the terrain and the host velocity [17]. Fade depths range up to 20 dB. The stochastic behavior of such transient link disconnections is very similar to that encountered for high frequency radio networks [18]. In the permanent case, a host is unable to communicate with a neighbor because their separation exceeds the range described by the cell geometry. We assume that each mobile host can communicate with an arbitrary mobile host in its cell without any interference (from other mobile hosts in the same cell) using techniques such as TDMA or code division multiple access (CDMA) spread spectrum signaling. One such technique is presented in [16].

3. BROADCAST CONSIDERATIONS. Terrestrial networks provide the means to manage RF spectrum utilization to minimize the inherent latency for transmission, the probability of detection, and the cost of utilization not afforded by satellite services. Reliable broadcast in a mobile wireless network is not easy due to the following reasons: (1) It may be difficult to maintain a convenient structure (spanning tree, virtual ring) for broadcasting because of the mobility of hosts and the absence of an established backbone network. While an adaptive algorithm can be used to maintain the structure, the small cell size leads to cases where two neighbors may frequently move out of each others' range leading to the invocation of the adaptive algorithm frequently. (2) The wireless network itself may not be connected always (see Figure 3). They may be decomposed into several connected components for a while and merge after some time (we assume "quite often"). We also assume that permanent disconnections do not occur. In the next section we present a preliminary solution for reliable broadcast in mobile wireless network. Our solution is based on simple (restricted) flooding and handshaking.

Flooding ultimately involves transmitting the message to every

node in the network, which is a disadvantage, particularly for large networks. The main advantage of flooding is that there is little explicit overhead and network management. As a consequence, no provisions are made to store or maintain routing or management data. Instead, hosts keep track of individual messages received and determine whether or not to retransmit the message. It is well suited to network requirements for highly mobile user groups on the digitized battlefield or in disaster relief operations where there is a need for reliable delivery in the presence of uncertain connectivity and rapid topology changes [19].

4. BROADCAST PROTOCOL. To broadcast a message, a mobile host transmits the message to all of its neighbors. On receiving a broadcast message, an intermediate mobile host retransmits the message to all of its neighbors. The technique would suffice if the network remained connected forever. Additional steps are necessary to cope with network link disconnections.

For example, mobile host h_i maintains a sequence counter, CNT_i . At any instant, the value of CNT_i denotes the number of messages broadcast by host h_i . CNT_i is incremented when h_i is about to broadcast a new message. In addition, host h_i maintains a history of messages ($HISTORY_i$) it has broadcast as well as received from other hosts. The j^{th} component of $HISTORY_i$ contains information about messages broadcast from h_i and received from h_j . A rebroadcast count and a time stamp provide the means to limit the propagation of the message in a large network.

A sample pictorial representation of $HISTORY_i$ is shown in Figure 4. Host h_j has received messages 1 and 4, but not

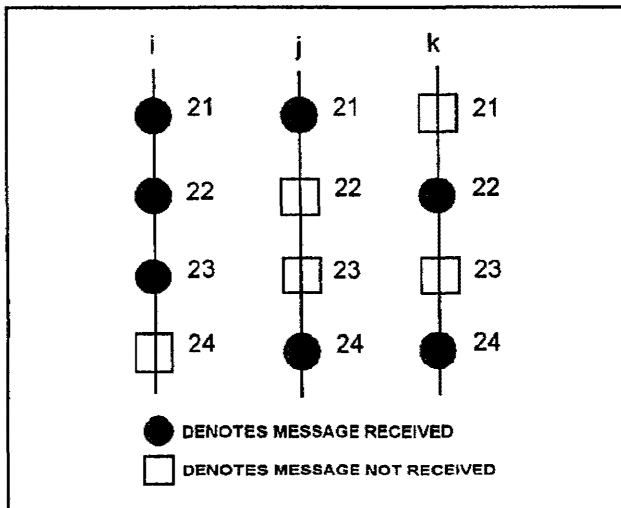


Figure 4. A snapshot of the status of the $HISTORY_i$ showing messages stored in h_i , h_j and h_k .

messages 2 and 3. Similarly, h_k has received messages 2 and 4 from h_j , but not messages 1 and 3. It is easy to maintain $HISTORY_i$ for each h_i as an array of lists. We now describe our solution for reliable broadcast.

Normal Operation. Let us assume that host h_i wants to broadcast message m . The following occurs:

- Host h_i first increments CNT_i and then transmits message (m, h_i, CNT_i) to all neighbors. It also stores (m, h_i, CNT_i) in a buffer locally.
- When host h_j receives message (m, h_i, CNT_i) , it sends an acknowledgment to the sender, updates the i^{th} component of $HISTORY_j$, and buffers the message locally. Host h_j then retransmits the message (m, h_i, CNT_i) to its neighbors.
- If h_j receives another copy of (m, h_i, CNT_i) , it discards the message, but sends an acknowledgment to the sender.
- A mobile host h , after sending a message (m, h_i, CNT_i) to its neighbors, waits for acknowledgments from all of its neighbors. If h does not receive acknowledgment from a neighbor within a certain time, h timeouts and resends the message (with a hope that link disconnection is transient). If h does not receive acknowledgment after several retries, h assumes that the link disconnection is not transient and stops sending the message.

During periods of heavy message exchange activity, this strategy substitutes for the polling or sounding procedure described in Section 2.

Handshake Procedure. When host h_j detects a new neighbor, h_k , a handshake procedure is executed by hosts h_j and h_k :

- h_j sends $HISTORY_j$ to h_k and receives $HISTORY_k$ from h_k .
- h_j compares $HISTORY_k$ with $HISTORY_j$ to identify messages available in $HISTORY_j$, but not in $HISTORY_k$, and broadcasts those messages. Host h_k does likewise.
- h_k then receives the "new" messages send by h_j and updates $HISTORY_k$. h_j does the same for messages received from h_k . also sends these "new" messages to other neighbors.

At the conclusion of the handshake procedure, the contents of $HISTORY_j$ and $HISTORY_k$ are equal. Thus, using handshake procedure, mobile hosts receive messages that they did not receive due to link disconnections. The size of $HISTORY$ stored at each h can be reduced as follows. If the first message received by h_j from h_k is $CNT_k=t$ then it is sufficient for the k^{th} component of $HISTORY_j$ to start from t . Storage for entry of messages 1 to $t-1$ need not be provided. Further optimization can be done by storing either the $HISTORY$ of messages received or the $HISTORY$ messages not received depending on which list is smaller.