EXHIBIT 67

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 68

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 69

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 70

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 71

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 72

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 73

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 74

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 75

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 76

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 77

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 78

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 79

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 80

Appl. No. 09/629,042 rney Docket No. 030048009US

Express Mail Label EV 335522411 US

#8/a

PATENT

[illegible]
9/24/03

OIPE JC112 SEP 10 2003 PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:	FRED B. HOLT *ET AL.*	EXAMINER:	BRADLEY E. EDELMAN
APPLICATION NO.:	09/629,042	ART UNIT:	2153
FILED:	JULY 31, 2000	CONF. NO:	4750
FOR:	**DISTRIBUTED GAME ENVIRONMENT**		

RECEIVED
SEP 1 5 2003
Technology Center 2100

Amendment Under 37 C.F.R. § 1.111

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Office Action dated May 21, 2003, please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of the Claims which begins on page 2 of this paper.

Amendments to the Drawings begin on page 6 of this paper and include attached drawing sheets.

Remarks/Arguments begin on page 7 of this paper.

09/12/2003 BSAYASI1 00000005 09629042
02 FC:1201 84.00 OP

A

Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently amended) A computer network for providing a game environment for 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, further wherein the network is m-regular, where m is the exact 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.

a1

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. (Cancelled)

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

A

5 ~~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.

6 ~~7~~. (Original) The computer network of claim 1 wherein all the participants are peers.

7 ~~8~~. (Original) The computer network of claim 1 wherein the connections are peer-to-peer connections.

A1 cont

8 ~~9~~. (Original) The computer network of claim 1 wherein the connections are TCP/IP connections.

9 ~~10~~. (Original) The computer network of claim 1 wherein each participant is a process executing on a computer.

10 ~~11~~. (Original) The computer network of claim 1 wherein a computer hosts more than one participant.

11 ~~12~~. (Original) The computer network of claim 1 wherein each participant sends to each of its neighbors only one copy of the data.

12 ~~13~~. (Original) The computer network of claim 1 wherein the interconnections of participants form a broadcast channel for a game of interest.

13
~~14~~. (Currently Amended) A distributed game system comprising:

a plurality of broadcast channels, each broadcast channel for playing a game, each of the broadcast channels for providing game information related to said game to 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 neighbor participants, further wherein the network is m-regular, where m is the exact 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;

a1 cont

means for identifying a broadcast channel for a game of interest; and

means for connecting to the identified broadcast channel.

14
~~15~~. (Original) The distributed game system of claim ~~14~~ 13 wherein means for identifying a game of interest includes accessing a web server that maps games to corresponding broadcast channel.

15
~~16~~. (Original) The distributed game system of claim ~~14~~ 13 wherein a broadcast channel is formed by player computers that are each interconnected to at least three other computers.

16
~~17~~. (New) A computer network for providing a game environment for a plurality of participants, each participant having connections to exactly four neighbor participants, wherein an originating participant sends data to the other participants by sending the data through each of its

a2

96

A

connections to its neighbor participants and wherein each participant sends data that it receives from a neighbor participant to its neighbor participants, further wherein the network is in a stable 4-regular state and wherein there are at least six participants to result in a non-complete graph.

17 ~~18.~~ (New) The computer network of Claim ~~17~~ 16 wherein a computer hosts more than one participant.

18 ~~19.~~ (New) A computer network for providing a game environment for 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, further wherein the network is m-regular and the network forms an incomplete graph.

C2 Cont

19 ~~20.~~ (New) The computer network of Claim ~~19~~ 18 wherein a computer hosts more than one participant.

127

A

Amendments to the Drawings

The attached sheets of drawings include changes to Figures 6 and 7. These sheets, which include Figures 6 and 7, replace the original sheets including Figures 6 and 7, respectively.

Attachment: Replacement Sheets

REMARKS/ARGUMENTS

Reconsideration and withdrawal of the rejections set forth in the Office Action dated May 21, 2003 are respectfully requested. In that Office Action, the Examiner objected to the drawings as failing to include certain reference signs mentioned in the description. Two replacement sheets for Figures 6 and 7 are submitted herewith with the appropriate reference signs included. The Examiner is requested to approve these replacement sheets for entry into this application.

Turning to the rejection of the claims based upon the prior art, the Examiner rejects Claims 14-16 under 35 U.S.C. § 102(b) being anticipated by Microsoft's Internet Gaming Zone; as well as being in public use more than one year prior to the filing date of this application as evidenced by the Internet Gaming Zone (IGZ) article. The Examiner also rejects Claims 1-13 as being obvious over the Alagar et al. paper.

The Cited Prior Art:

The IGZ article is a press release detailing the Internet Gaming Zone by Microsoft. As detailed in the press release, the IGZ article describes a system that allows for multi-player gaming via the Internet. There is however no indication as to how such a network system is implemented.

The Alagar reference relates to a reliable mobile wireless network. The term "mobile wireless network" as used in Alagar means that the network does not contain any static support stations. The example given in the Alagar reference is of a military theater where each of the nodes (troops, tanks, etc. . . .) are mobile and can communicate with each other using wireless transmissions. Because of the mobile nature of the network, there are frequent changes in link connectivity between various nodes. The mobile wireless network, because it does not contain any static support stations, is dissimilar to the Internet or even cellular telephony.

A

Because of the mobile nature of the network nodes, the Alagar reference teaches that two mobile nodes are "neighbors" if they can hear each other. Each host detects its neighbors by periodically broadcasting a probe message. A host that hears a probe message sends an acknowledgement to the probing host. Every host maintains a list of neighbors and periodically updates the list based on acknowledgements 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 update their list of neighbors.

Because of the mobile nature of the nodes, it is not uncommon that the link may be disconnected between two nodes. Because of this, messages are transmitted from node to node using a flooding methodology that involves transmitting the message to every node in the network. Thus, to broadcast a message, a mobile node 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 Alagar reference also provides a methodology for limiting the amount of retransmission of messages. This is accomplished by means of an acknowledgement protocol.

The Examiner's Arguments:

The Examiner rejects Claims 14-16 under 35 U.S.C. § 102 as being anticipated by the IGZ article. The Examiner argues that the IGZ article discloses a plurality of broadcast channels and means for broadcasting a broadcast channel for topics of interest.

Next, the Examiner rejects Claims 1-13 under 35 U.S.C. § 103 as being obvious over the Alagar et al. reference. The Examiner argues that Alagar discloses a plurality of nodes that form a network and that the data is sent to the other participants by a flooding technique.

Applicants respectfully request reconsideration.

Applicants' Amendments and Arguments:

Applicants have significantly amended independent Claims 1 and 14. In addition, new independent Claims 17 and 19 have been added which applicants believe should be allowable over the cited prior art in view of the remarks set forth below. In view of the substantial amendments made to Claim 14 to include all of the limitations of Claim 1, the arguments will be primarily directed towards the Alagar reference which was used to reject Claims 1-13.

First, one important aspect of the Alagar reference is that the flooding protocol disclosed in Alagar dictates that when a node receives a message, that node will rebroadcast that message to **all** of its neighbors. See Alagar at page 239, column 1, lines 13-15. Specifically, the Alagar reference at page 239, column 2, lines 7-23 dictates that whenever a host (i.e., node) receives a message, that message is broadcast to all of its neighbors.

In contrast, the present claimed invention of Claim 1 dictates and requires that each participant only rebroadcasts received messages to its neighbors **other than** the neighbor from which the node received the message. The Alagar reference requires a larger number of messages to be broadcast. For example, if m is the number of nodes and N is the number of neighbors for each node, then the total number of messages is m x N.

In contrast, by limiting the rebroadcast to "other neighbors," this reduces the number of messages to be broadcast to (m-1)N + 1. For large networks, the saved bandwidth can be significant. For this sole reason alone, Claim 1 has a requirement of "other neighbors" which is not fairly shown in the Alagar reference. Therefore, Claim 1 and all dependent claims therefrom are in condition for allowance.

Secondly, the Alagar reference teaches the indiscriminant linking with neighbors regardless of the number of total neighbors that are capable of being connected. For example, Alagar teaches that the definition of a "neighbor" is any two mobile hosts that can "hear" each other. See Alagar at page 238, column 1, lines 5-6. In other words, there is no "regularity" to the network formed by Alagar because each of the nodes can link to as few as one neighbor or a potentially extremely large number of neighbors. The only limitation is that the node will link and classify as a neighbor any other node that is within hearing distance. This is precisely the opposite of the amended claimed invention. Claim 1 as amended requires that each participant in the network connects to and forms a neighbor bond to exactly an m number of neighbors. Independent claims 14 and 17 contain similar limitations.

Figure 1 of the Alagar reference is deceiving in that it coincidentally shows a 4-regular network. However, that is not the typical situation as is clear from a careful review of the Alagar reference. Column 1 of page 238 of the Alagar reference clearly indicates that there is in fact nonregularity in a computer network formed because the number of neighbors is not set at a predetermined number, but rather based upon the particular encountered terrain of the mobile nodes.

Claim 1 as amended requires that the computer network be m regular at substantially all times where there are not new nodes entering or leaving the network. Furthermore, Claim 17 requires that the network is "in a stable 4-regular state." For this reason, the claims are allowable over the cited prior art.

Third, and yet another independent reason for allowing the claims, as amended, over the Alagar patent, is that the claims as amended now require that the computer network so formed is not a "complete graph." A complete graph is a network that is characterized by $N = m + 1$. A "complete graph" in graph theory is that each node has a connection to every other node in the network. Thus, Figure 1 of the Alagar reference shows a complete graph. Each of the nodes has a connection to every other node in the network. Obviously, for a five-node network, this will require four communications connections for each node.

Claims 1 and 17 have been amended to recite that there are at least two more nodes than there are maximum number of neighbors. For example, Claim 17 requires that for a 4-regular network, there are at least six participants. Claim 1 requires that the parameter N is at least two greater than the parameter m. Alagar does not show this limitation whatsoever. In fact, the only m-regular network shown in Alagar is a complete graph. It is the combination of having a computer network that is m regular and that is not a complete graph that is patentable over the Alagar reference. This combination has been shown to produce an efficient and stable computer network. Claim 19 is specifically directed to this aspect of the invention.

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 prior 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) 359-6488.

Respectfully submitted,
Perkins Coie LLP

[illegible]

Chun M. Ng
Registration No. 36,878

Date: 9/10/03

Correspondence Address:
Customer No. 25096
Perkins Coie LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000

EXHIBIT 81

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 82

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 83

US007596633B2

(12) **United States Patent**
Mai et al.

(10) **Patent No.: US 7,596,633 B2**
(45) **Date of Patent: Sep. 29, 2009**

(54) **ISLAND RECOVERY IN A PEER-TO-PEER RELAY NETWORK**

(75) Inventors: **Anthony Mai**, San Marcos, CA (US); **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Forster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 857 days.

(21) Appl. No.: **10/701,298**

(22) Filed: **Nov. 3, 2003**

(65) **Prior Publication Data**

US 2005/0086369 A1 Apr. 21, 2005

Related U.S. Application Data

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/249**; 709/224; 709/244; 370/216

(58) **Field of Classification Search** 709/228, 709/237, 238, 224, 227, 244, 249; 370/216, 370/230

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,590,865	B1 *	7/2003	Ibaraki et al.	370/230
6,920,501	B2 *	7/2005	Chu et al.	709/228
7,177,951	B1 *	2/2007	Dykeman et al.	709/249
7,194,654	B2 *	3/2007	Wray et al.	370/216
2001/0044339	A1	11/2001	Cordero et al.	
2001/0046213	A1 *	11/2001	Sakoda	370/328
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1	12/2002	Traversat et al.	
2003/0055892	A1	3/2003	Huitema et al.	
2005/0007964	A1 *	1/2005	Falco et al.	370/256

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999
EP	1 107 508	6/2001

(Continued)

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5. Dec. 1, 2003, pp. 2891-2895, XP010678188.

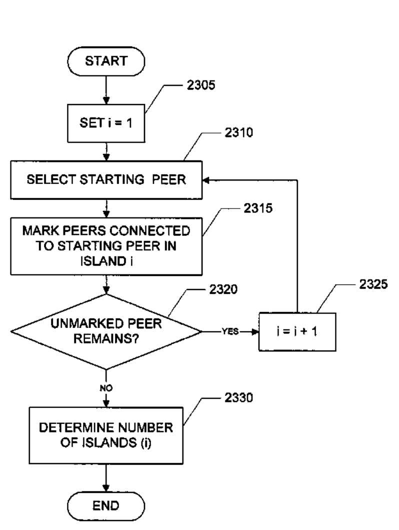
(Continued)

Primary Examiner—Ramy Mohamed Osman
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Thomas F. Presson

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of island recovery in a peer system in a peer-to-peer relay network includes: detecting the presence of two or more islands in a peer-to-peer relay network, wherein each island includes at least one peer system; joining two detected islands by connecting a peer system in a first island to a peer system in a second island; wherein peer systems in different islands are not connected.

16 Claims, 31 Drawing Sheets



FOREIGN PATENT DOCUMENTS

WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

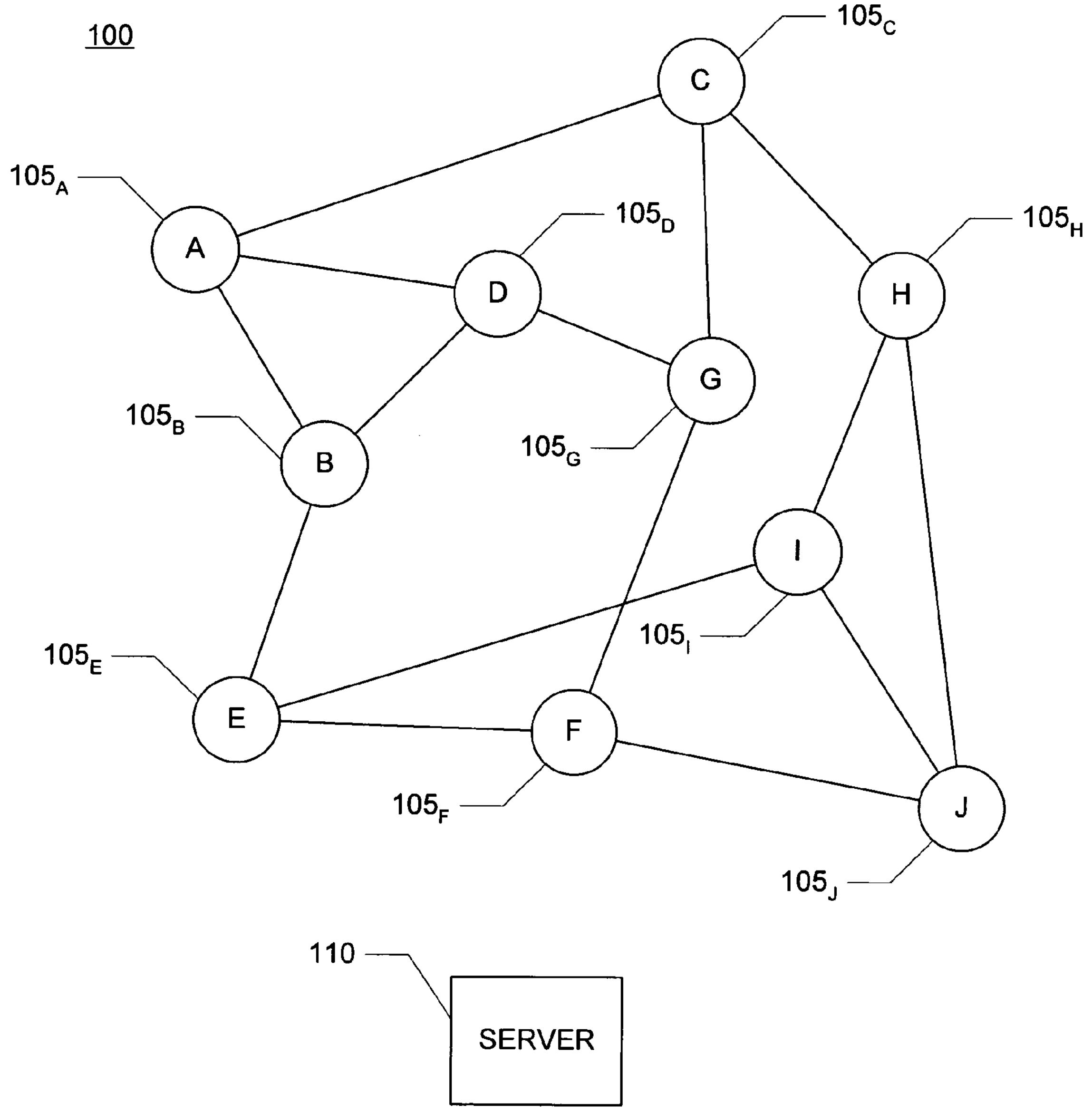
OTHER PUBLICATIONS

Dutkiewicz E Ed —Institute of Electrical and Electronics Engineers: “Impact of transmit range on throughput performance in mobile ad hoc networks” ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed —Association for Computing Machinery: “Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups” Proceedings of the 7^{TH} ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000. ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. CONF. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113 -203-4.

* cited by examiner

FIG. 1



100
A
B
C
D
E
F
G
H
I
J
105A
105B
105C
105D
105E
105F
105G
105H
105I
105J
110
SERVER

FIG. 2

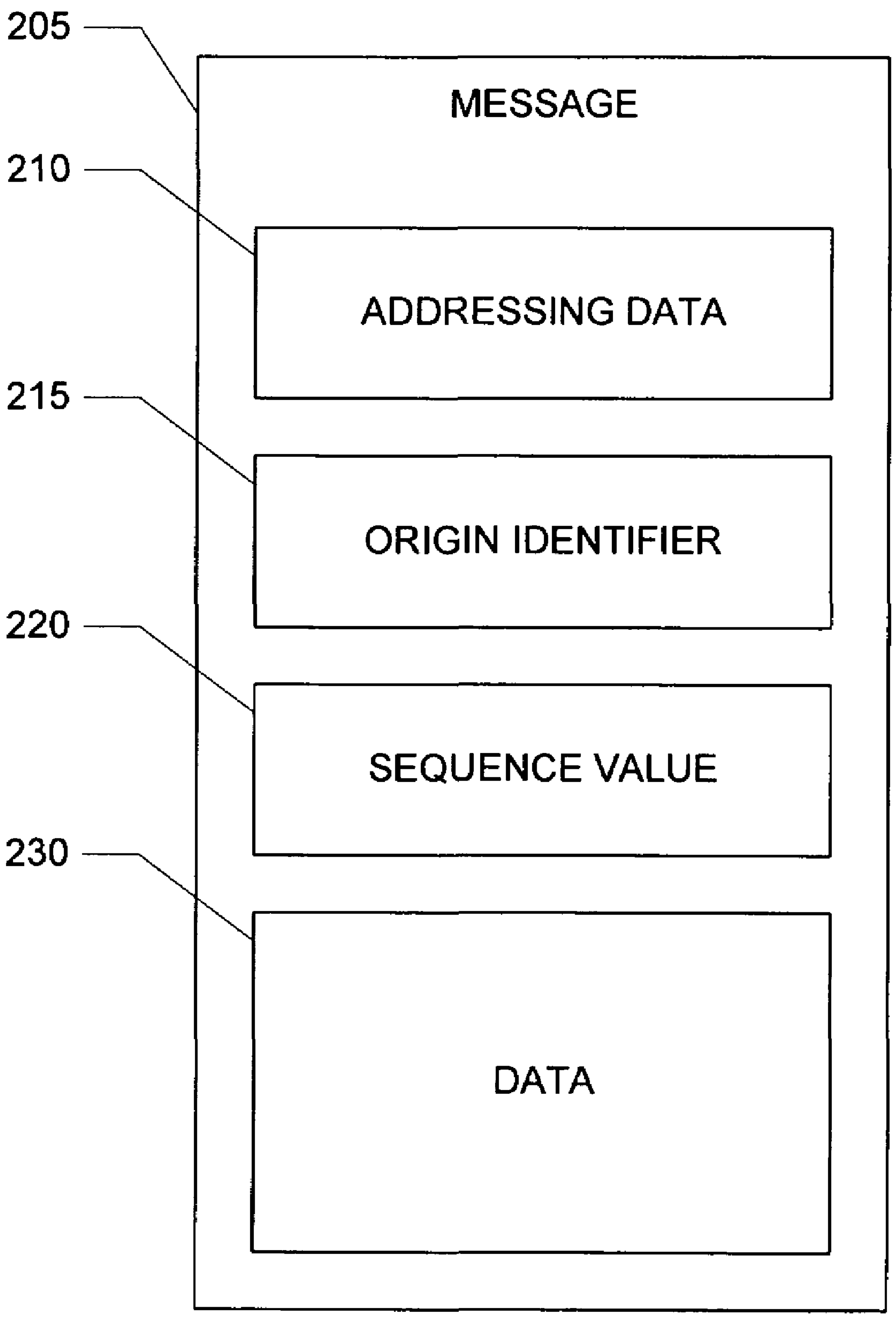

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

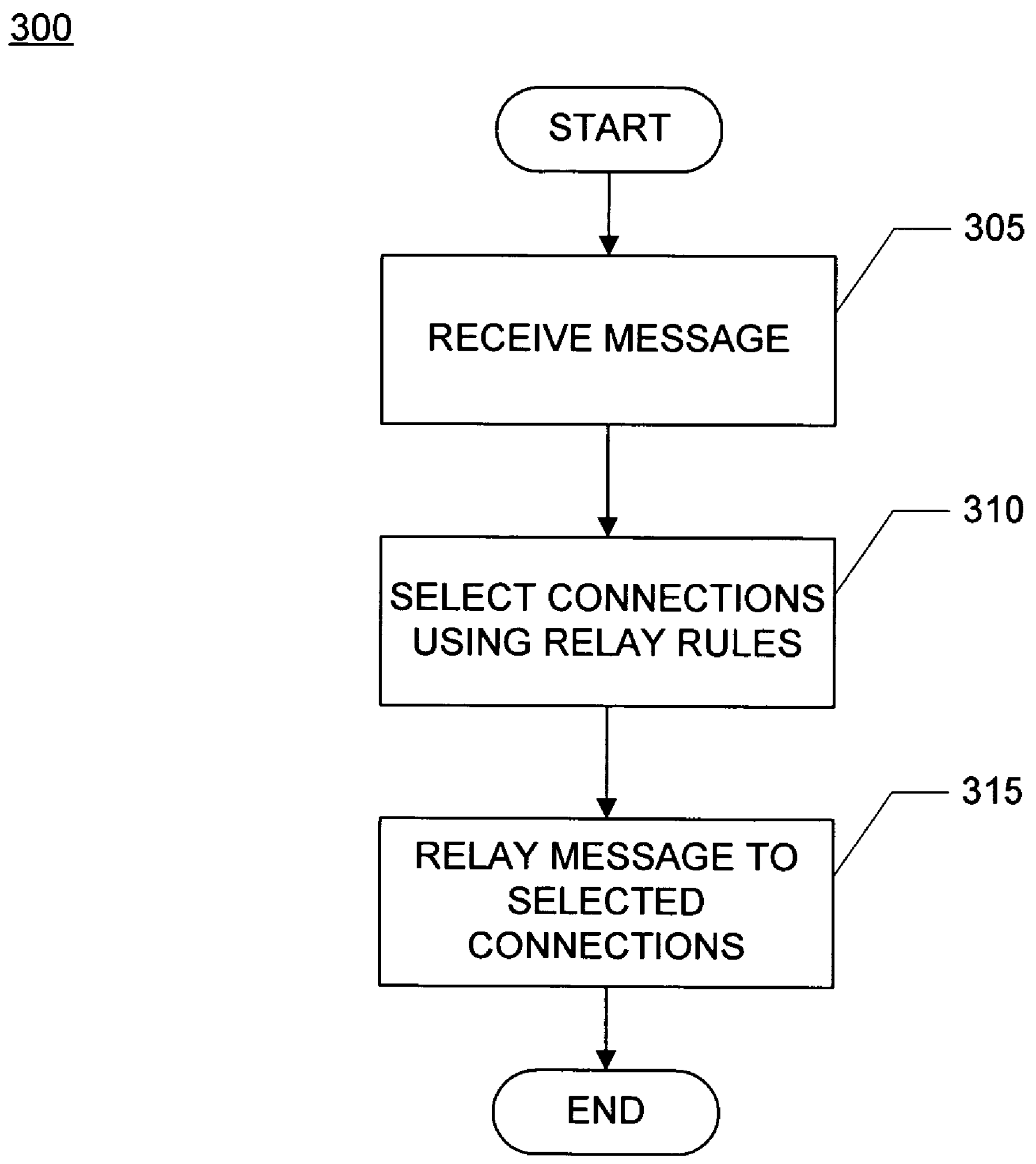

300
START
RECEIVE MESSAGE
305
SELECT CONNECTIONS USING RELAY RULES
310
RELAY MESSAGE TO SELECTED CONNECTIONS
315
END

FIG. 4

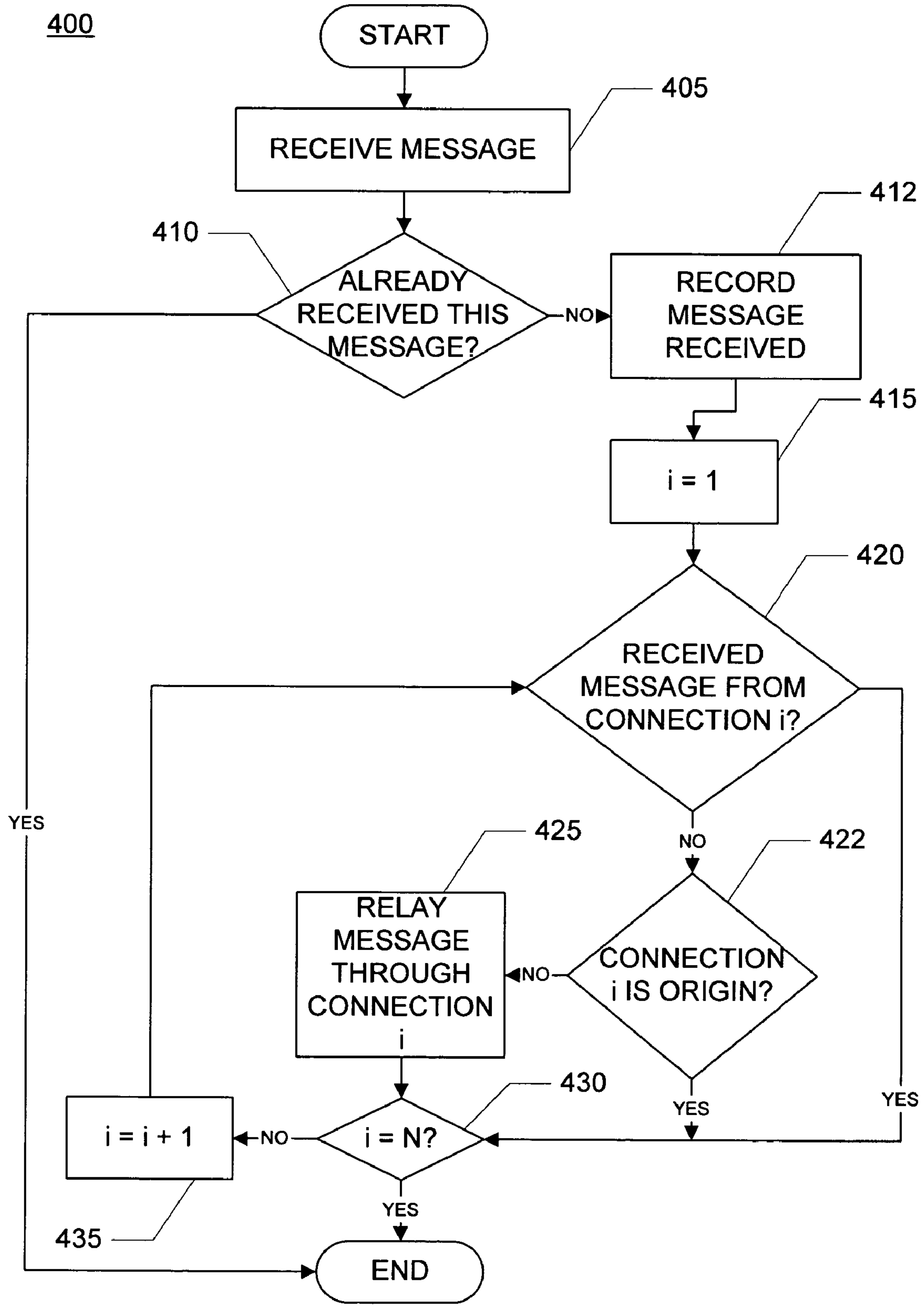

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
RECORD MESSAGE RECEIVED
412
415
i = 1
420
RECEIVED MESSAGE FROM CONNECTION i?
YES
NO
425
422
RELAY MESSAGE THROUGH CONNECTION i
NO
CONNECTION i IS ORIGIN?
430
YES
YES
i = i + 1
NO
i = N?
YES
435
END

FIG. 5

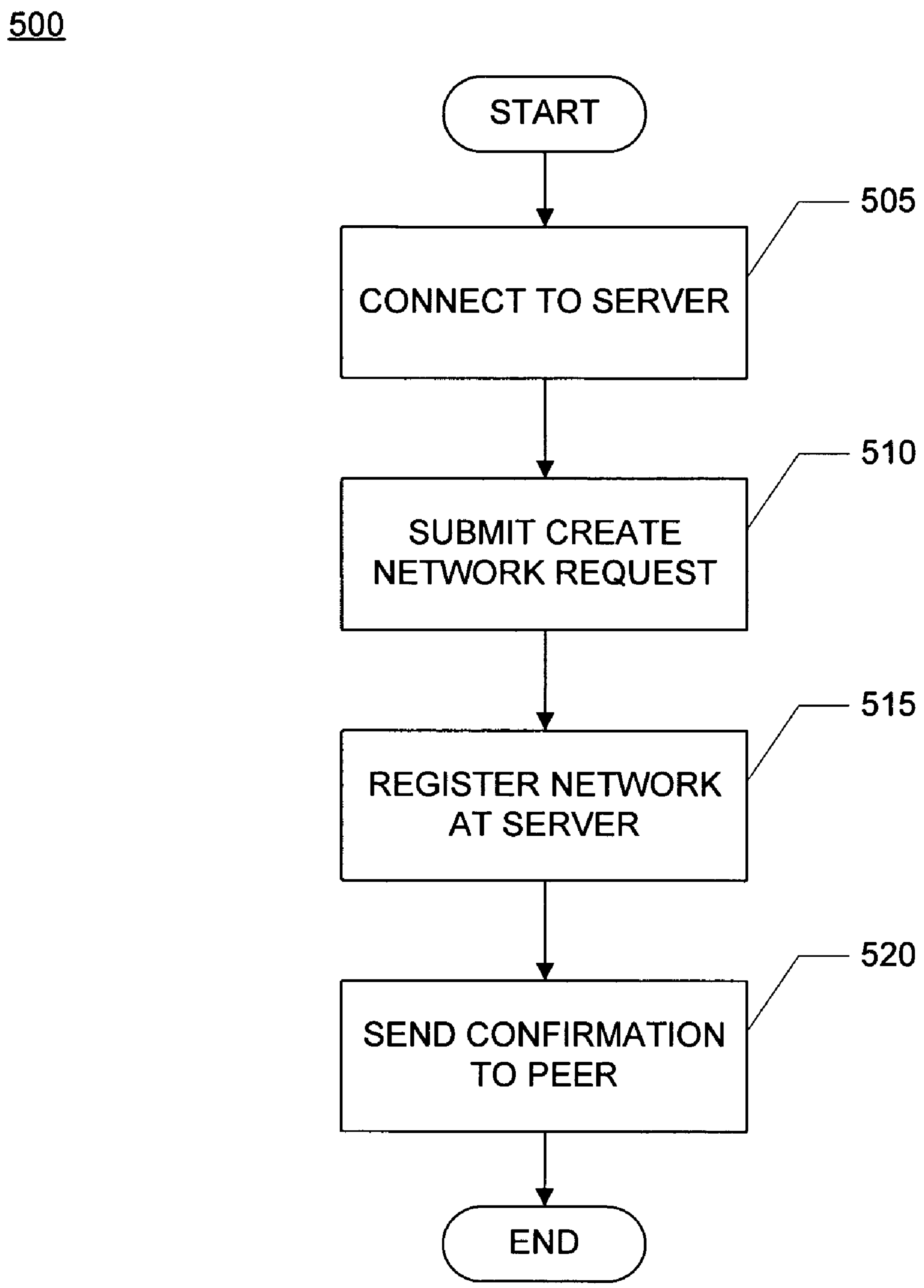

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

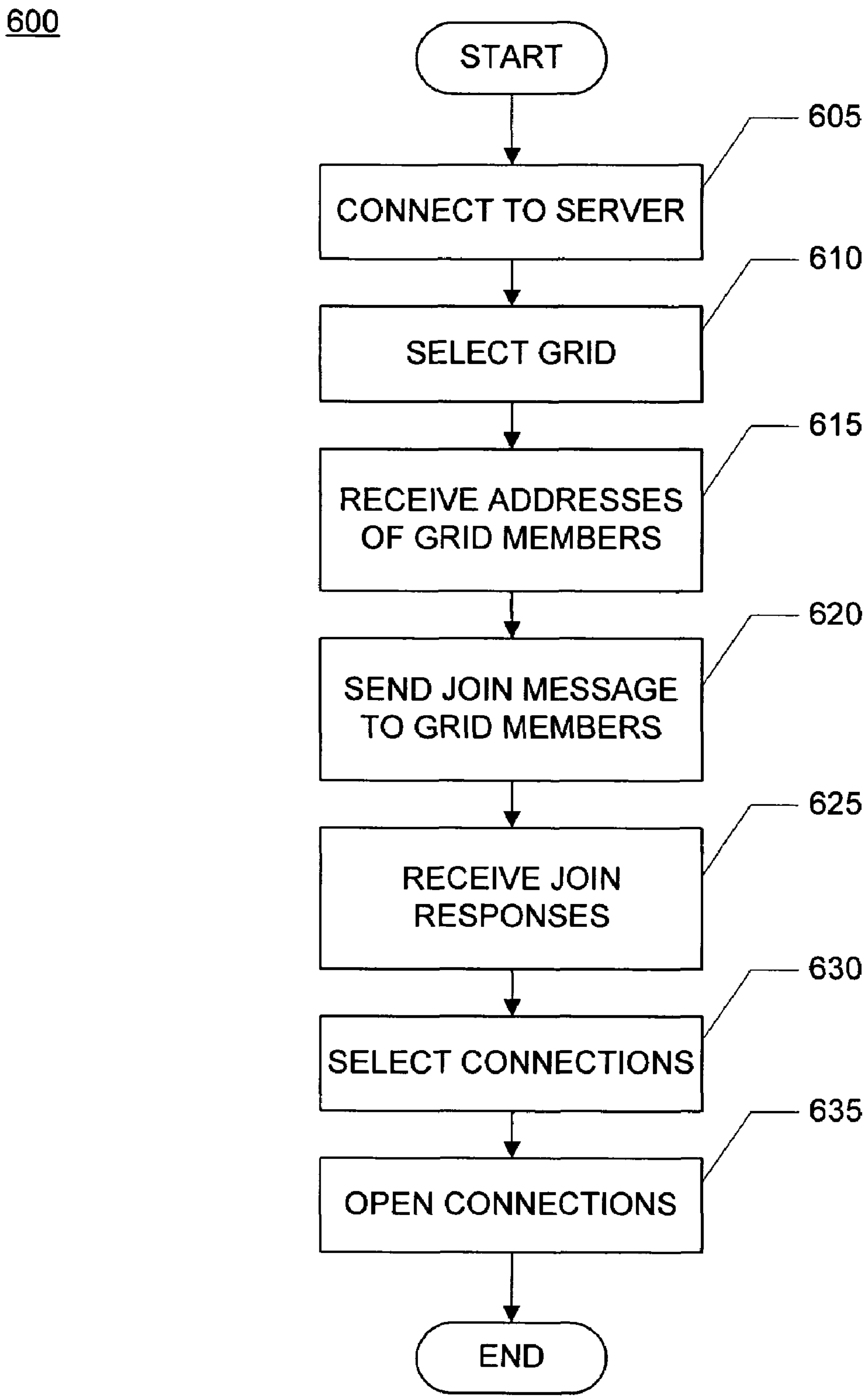

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

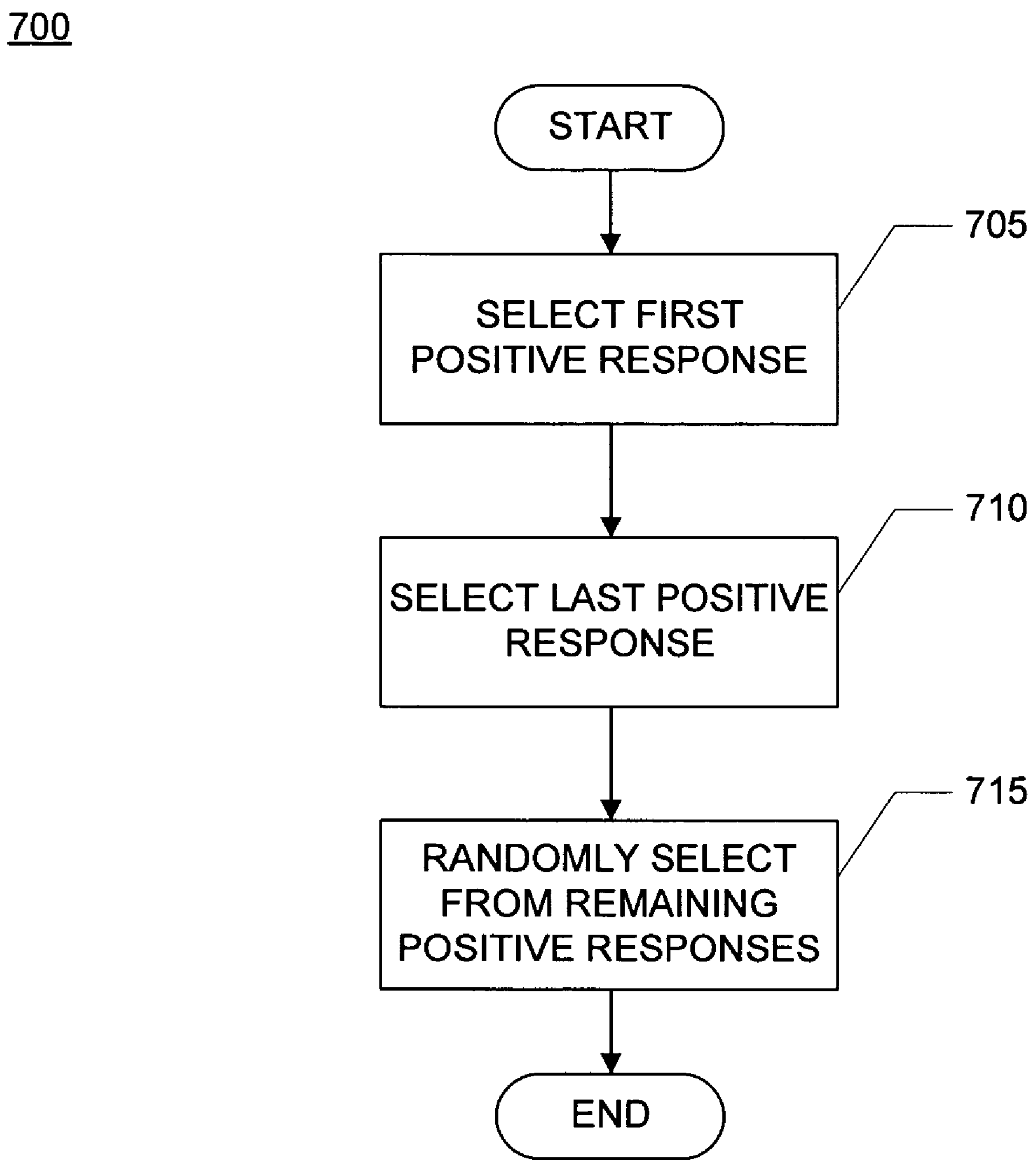

700
START
705
SELECT FIRST POSITIVE RESPONSE
710
SELECT LAST POSITIVE RESPONSE
715
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
END

FIG. 8

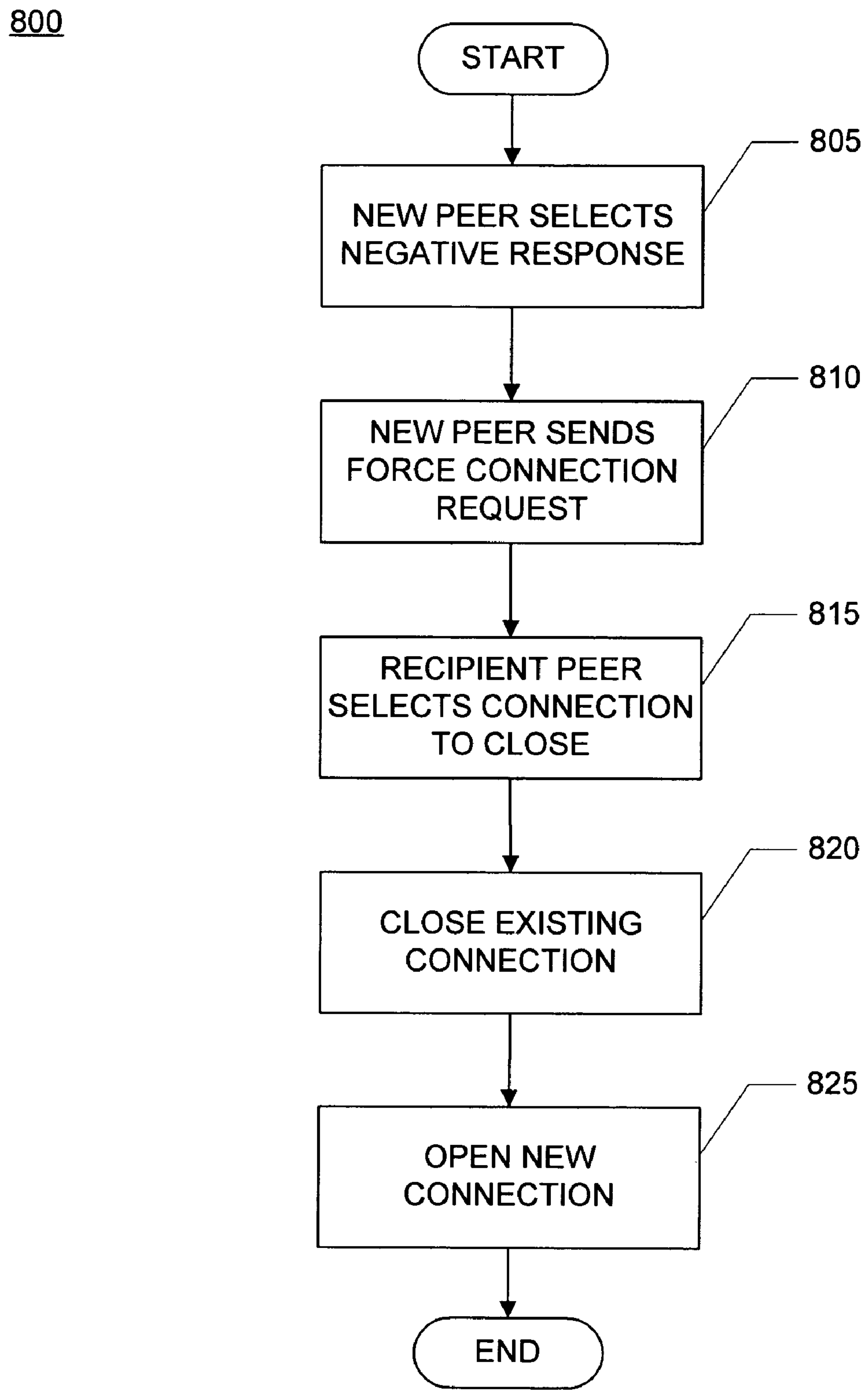

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

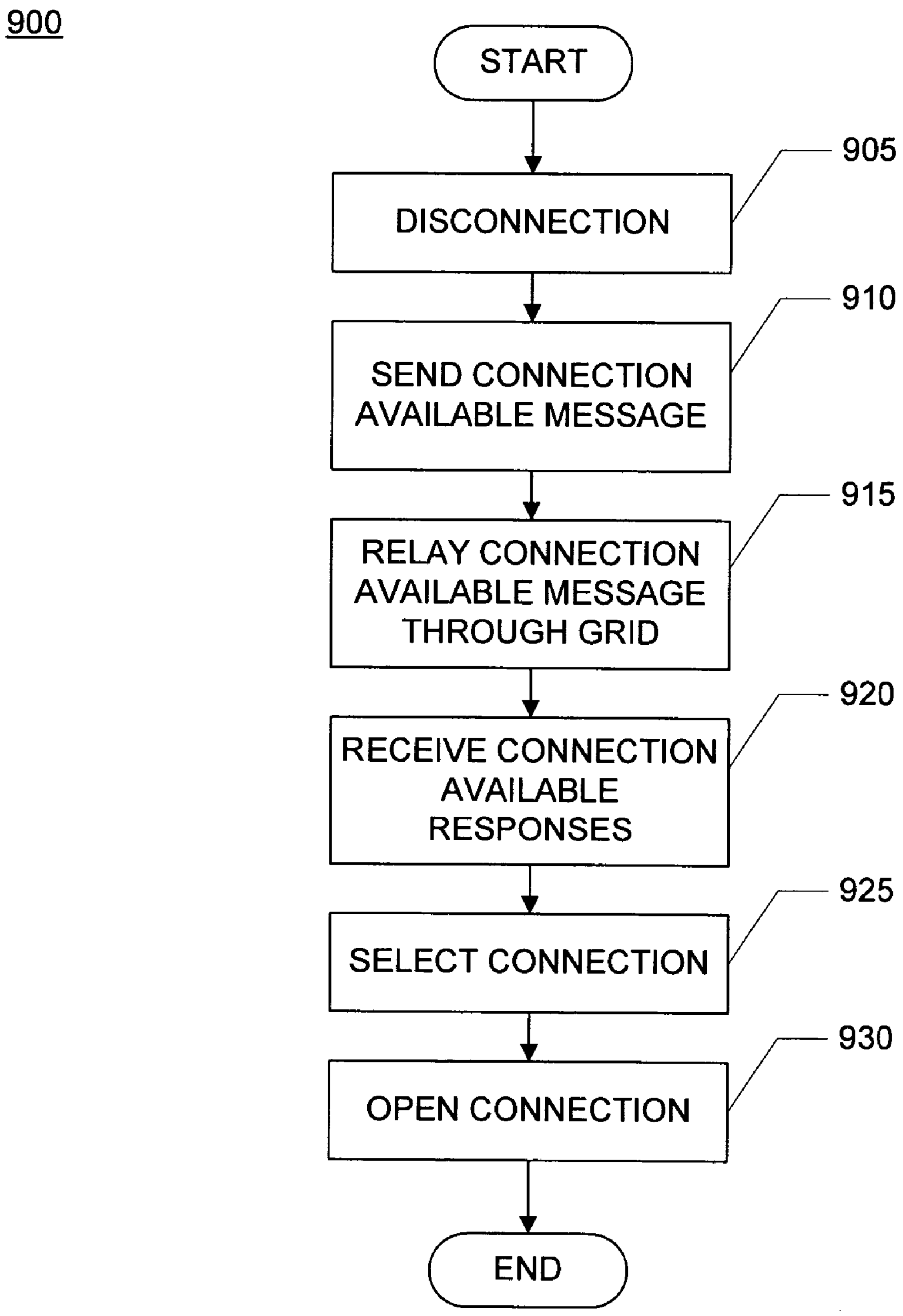

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

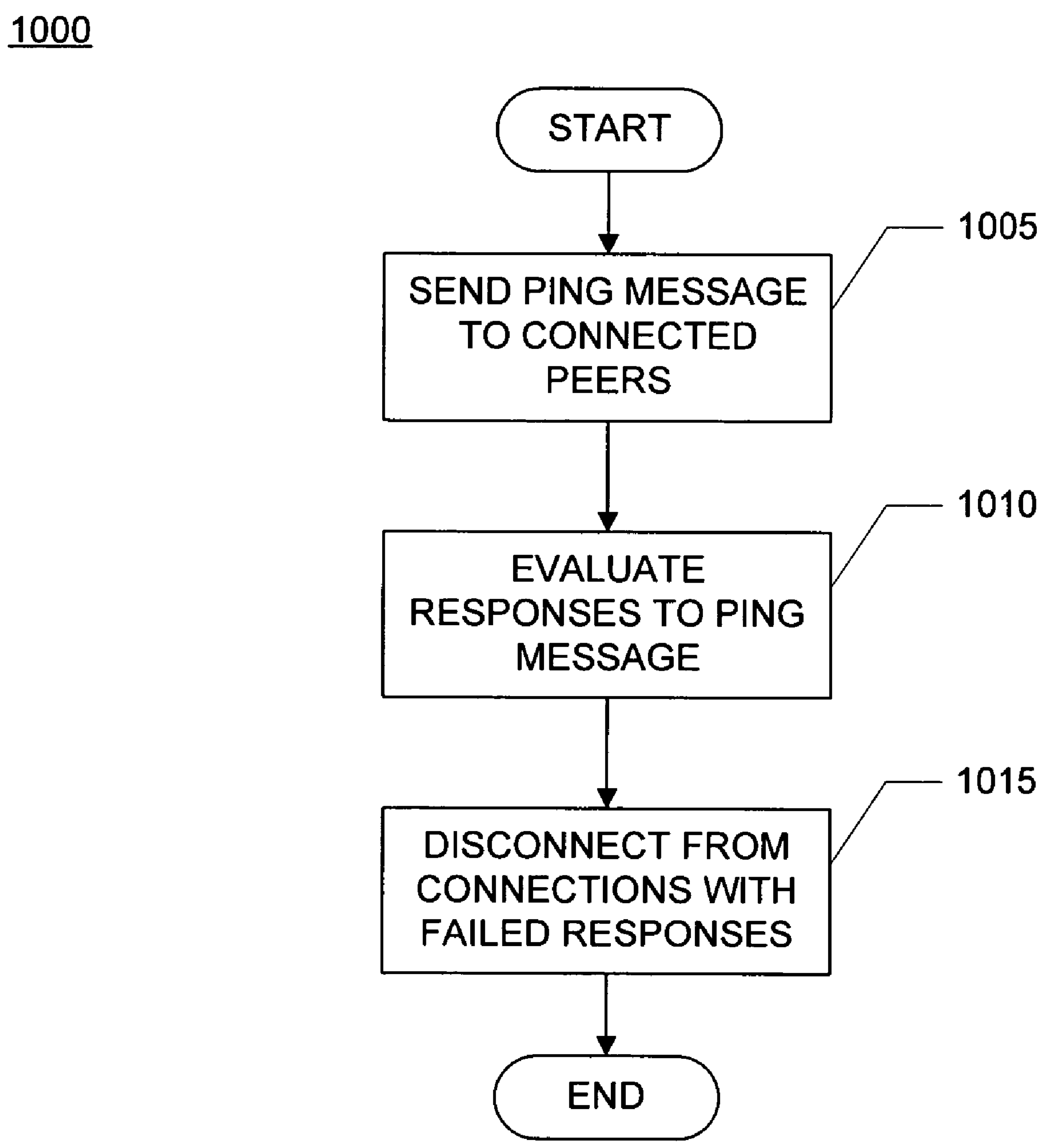

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

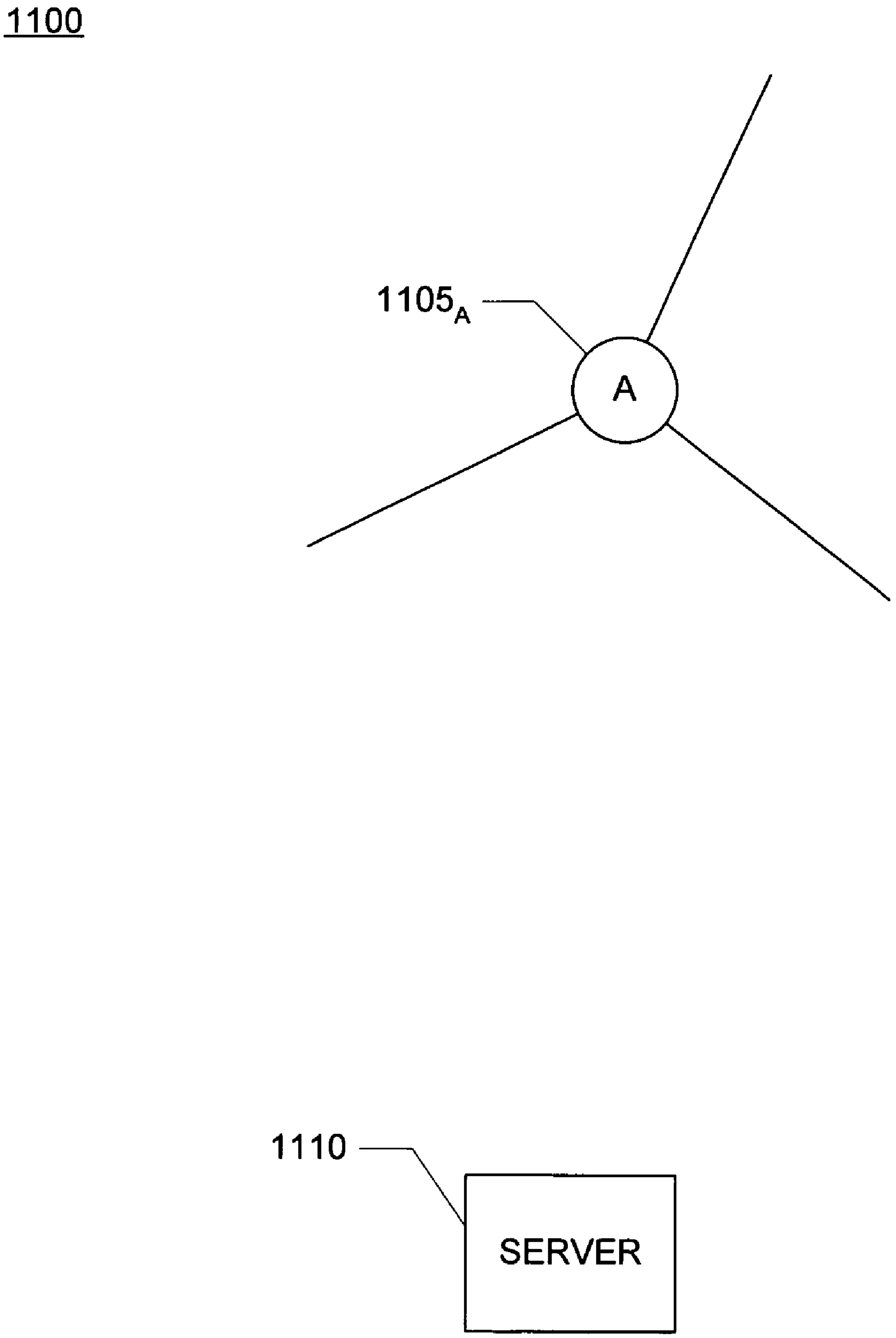


FIG. 12

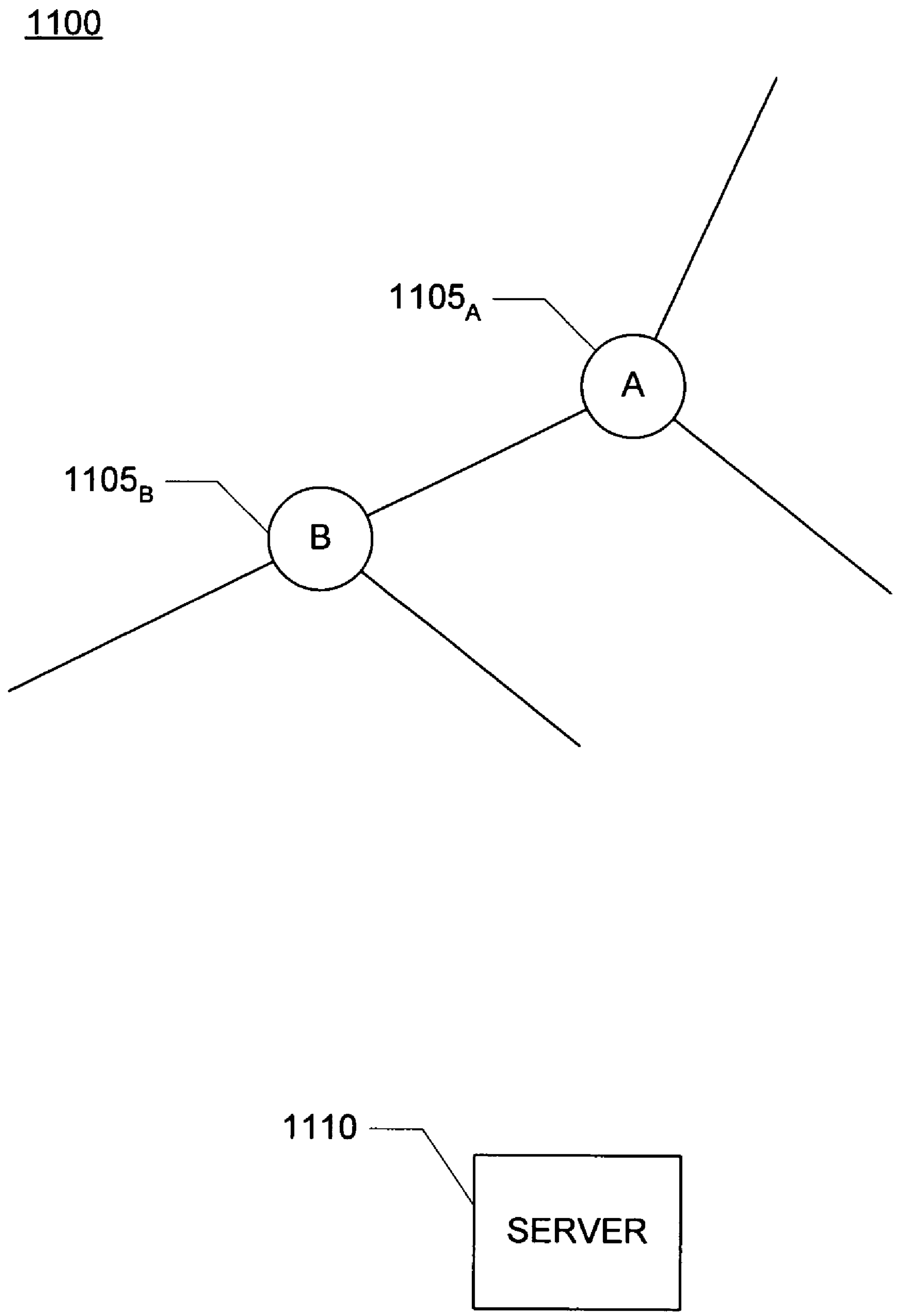


FIG. 13

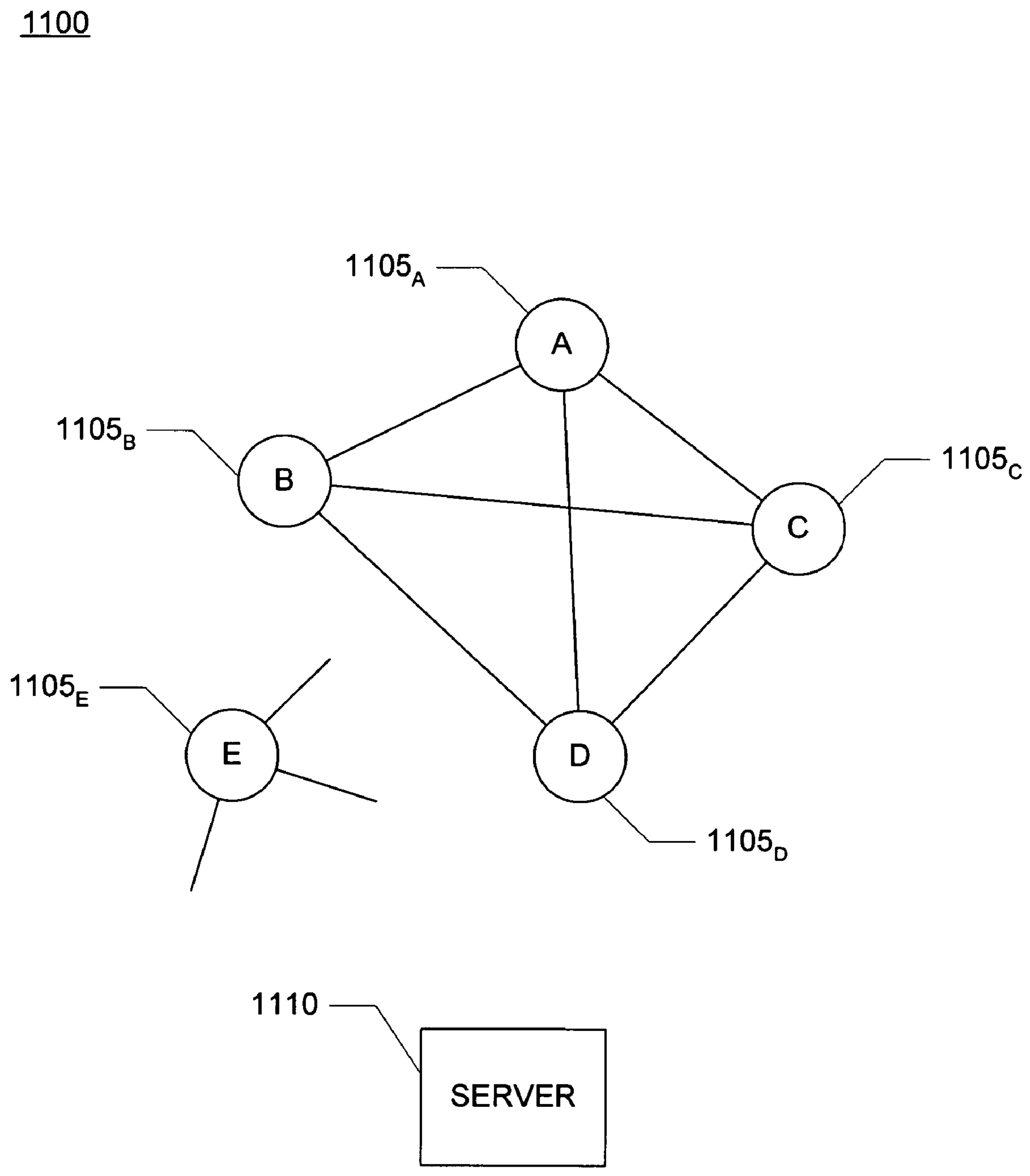


FIG. 14

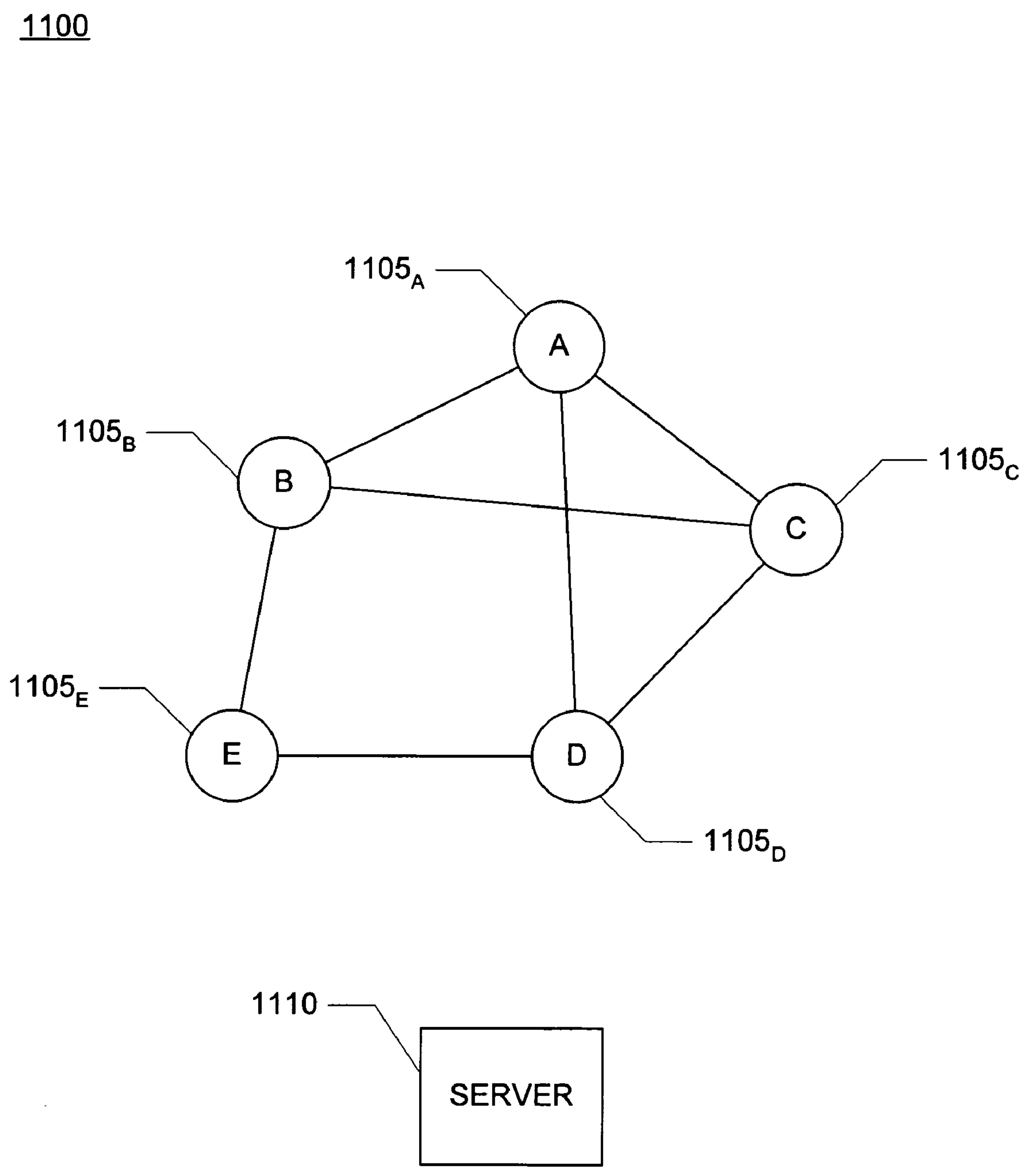


FIG. 15

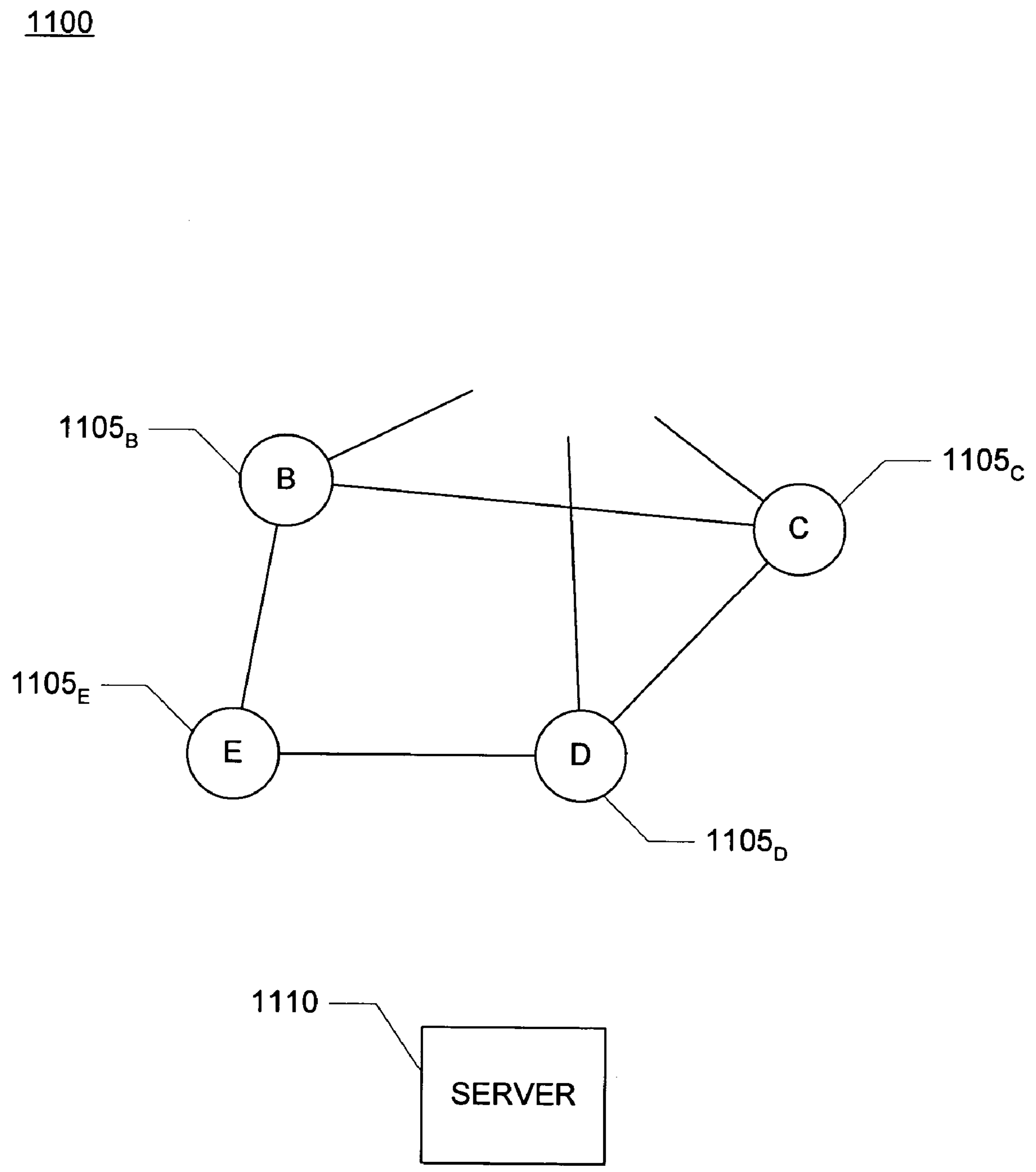


FIG. 16

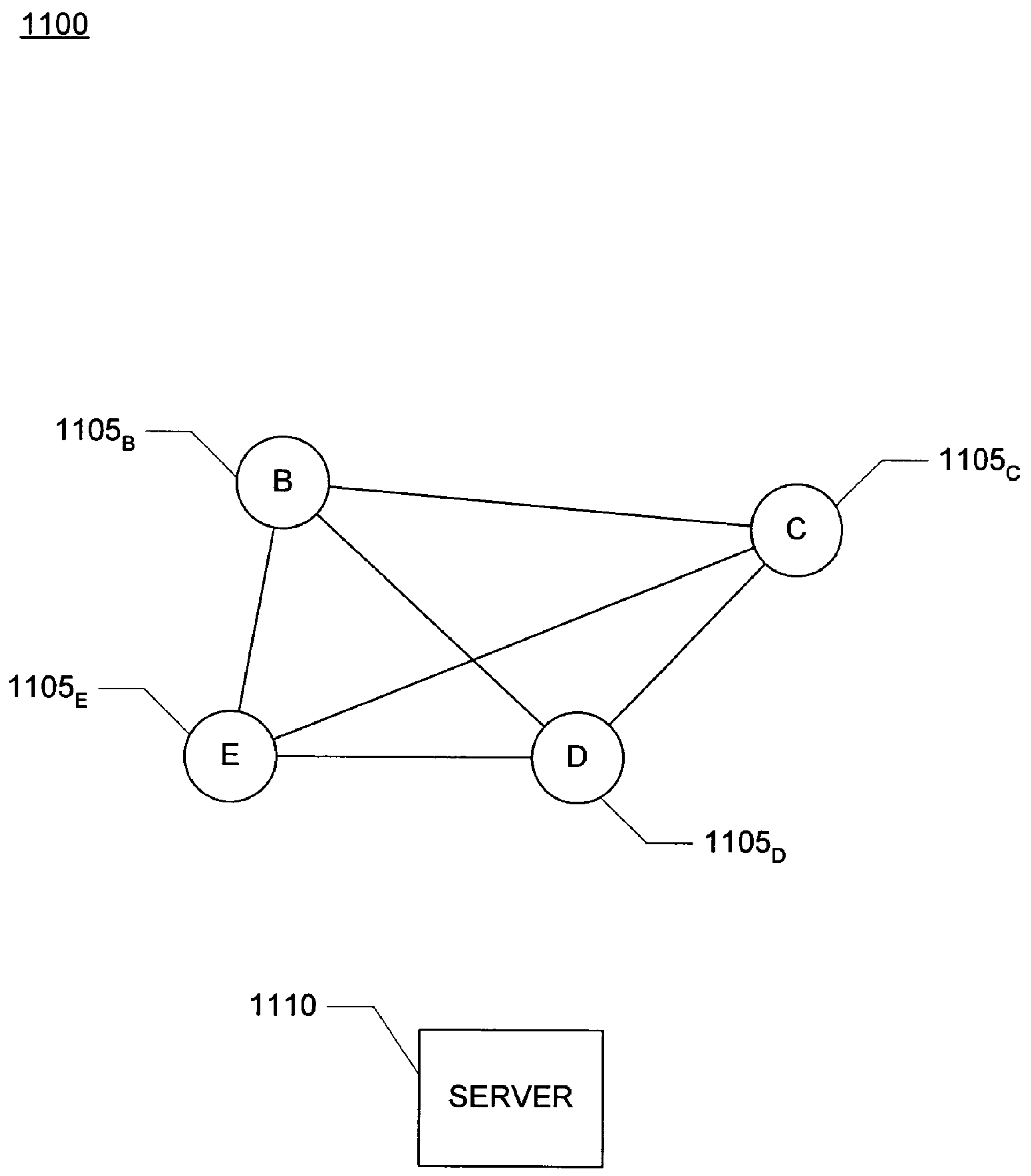


FIG. 17

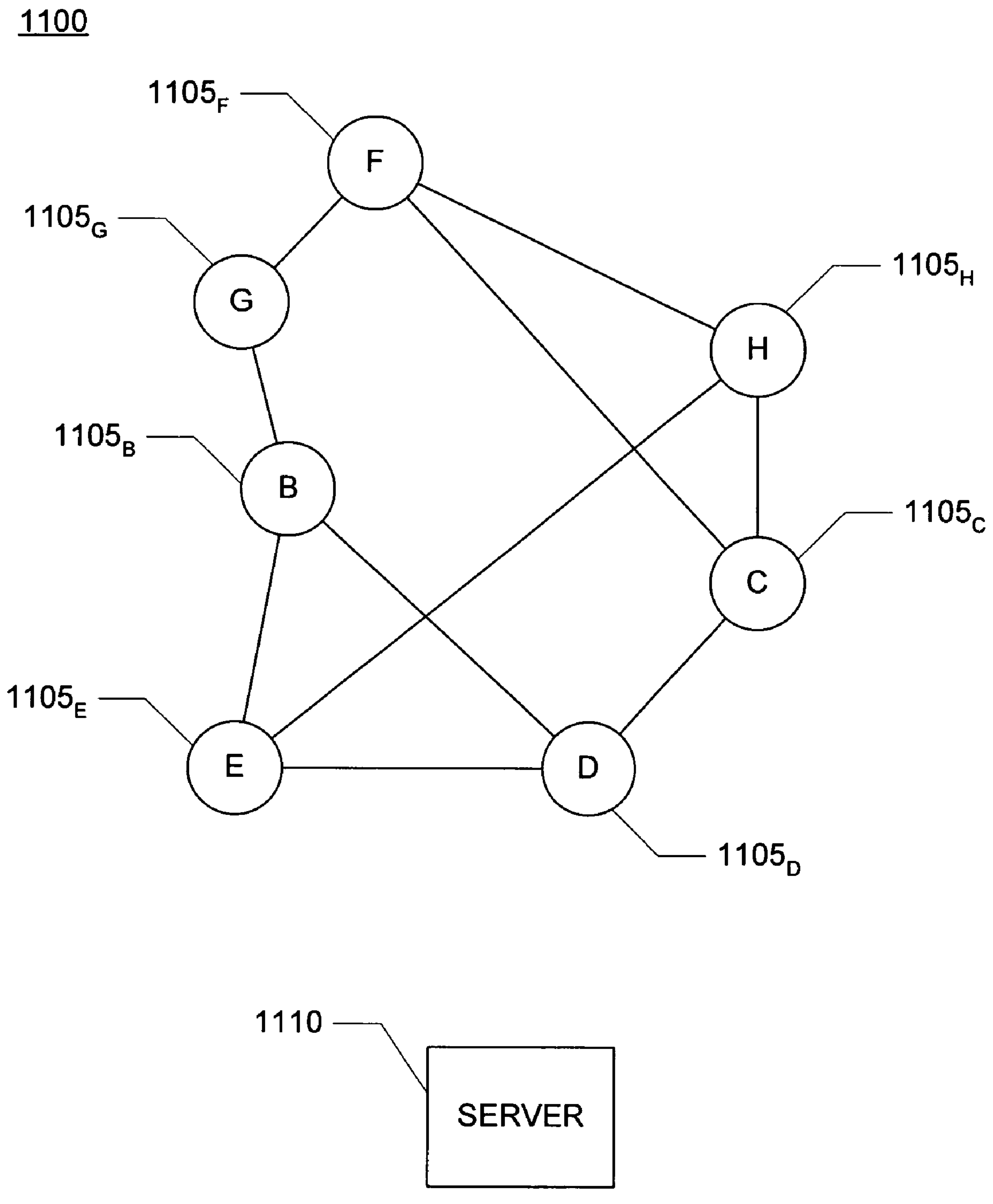

1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 18

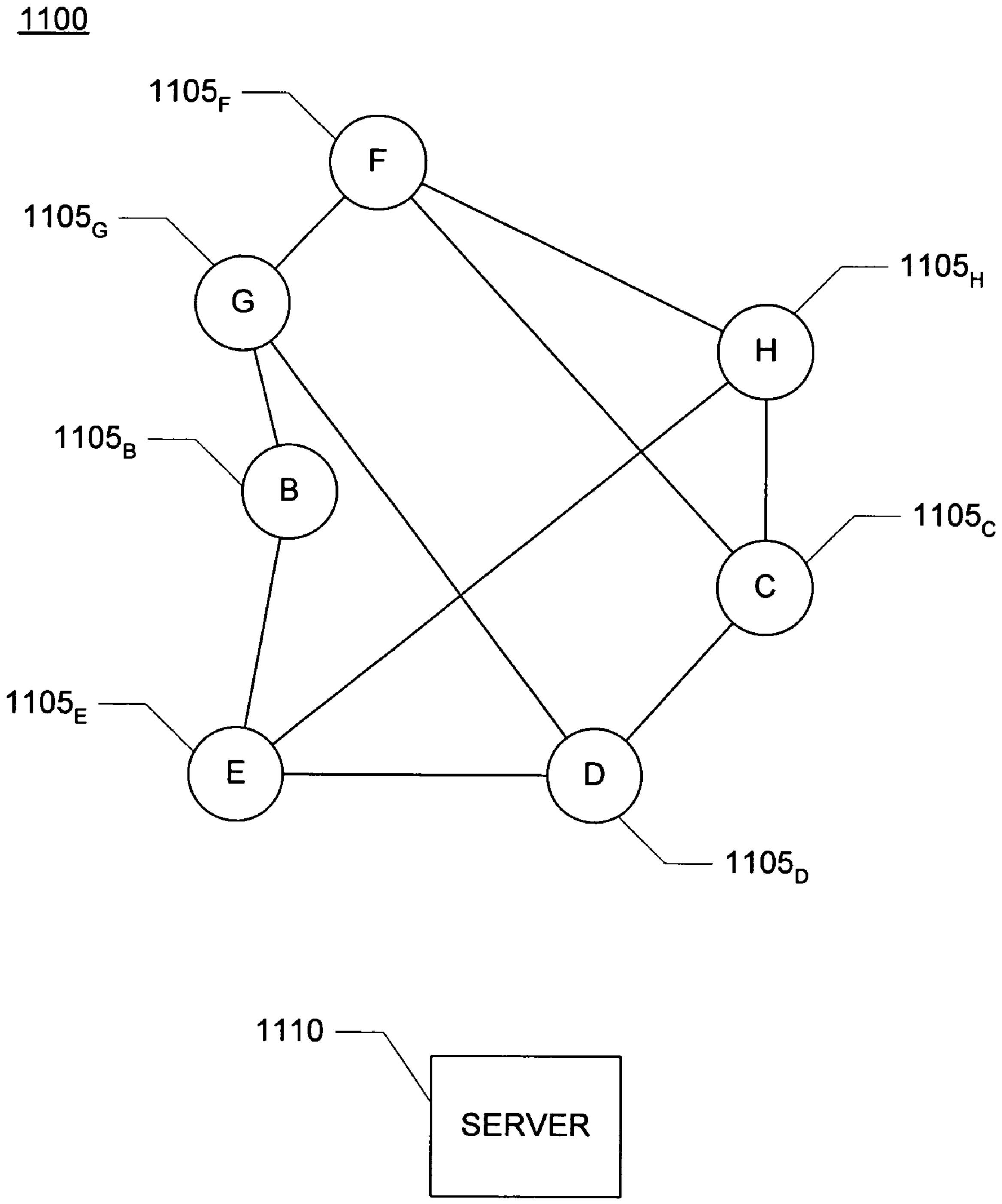

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 19

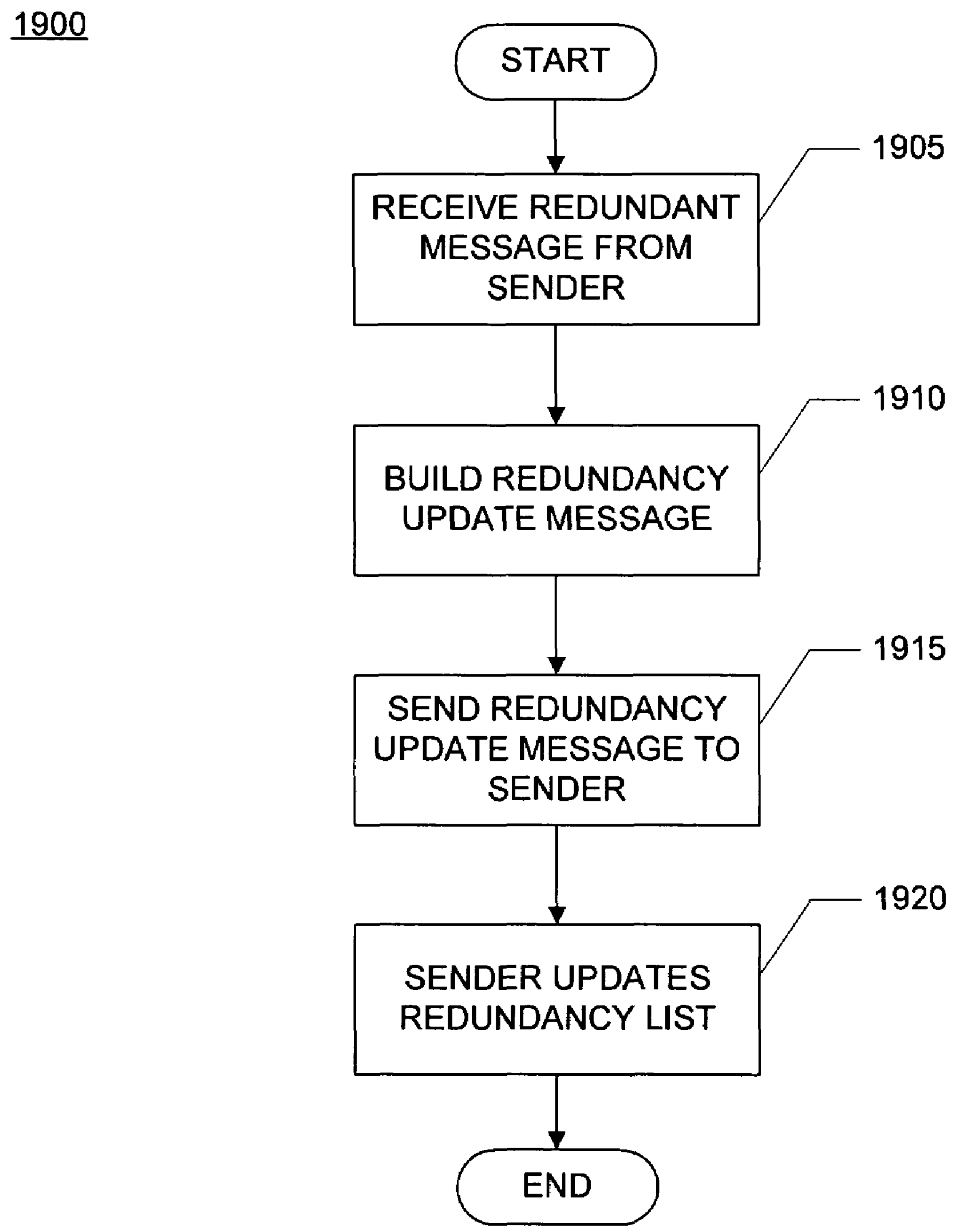

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

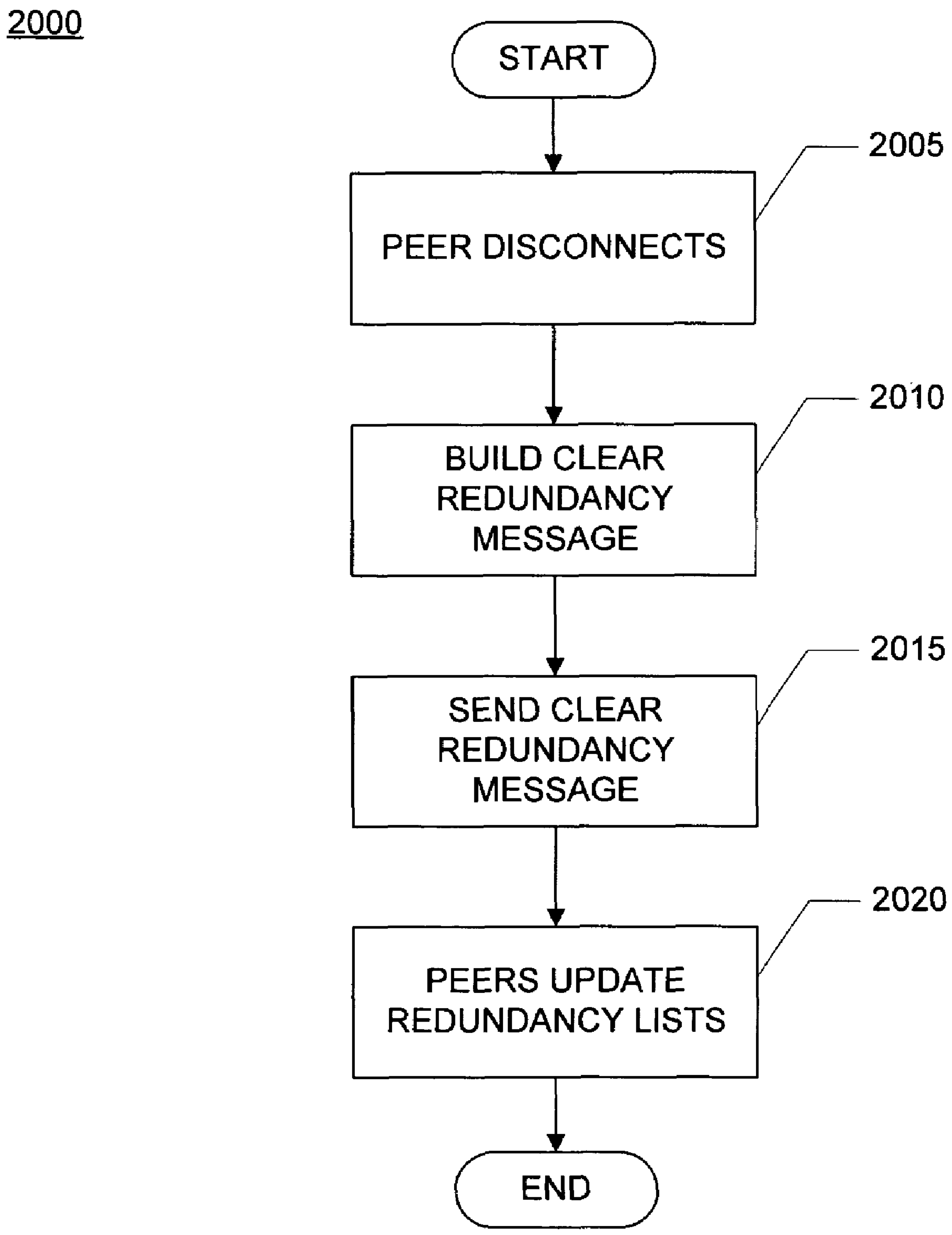

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

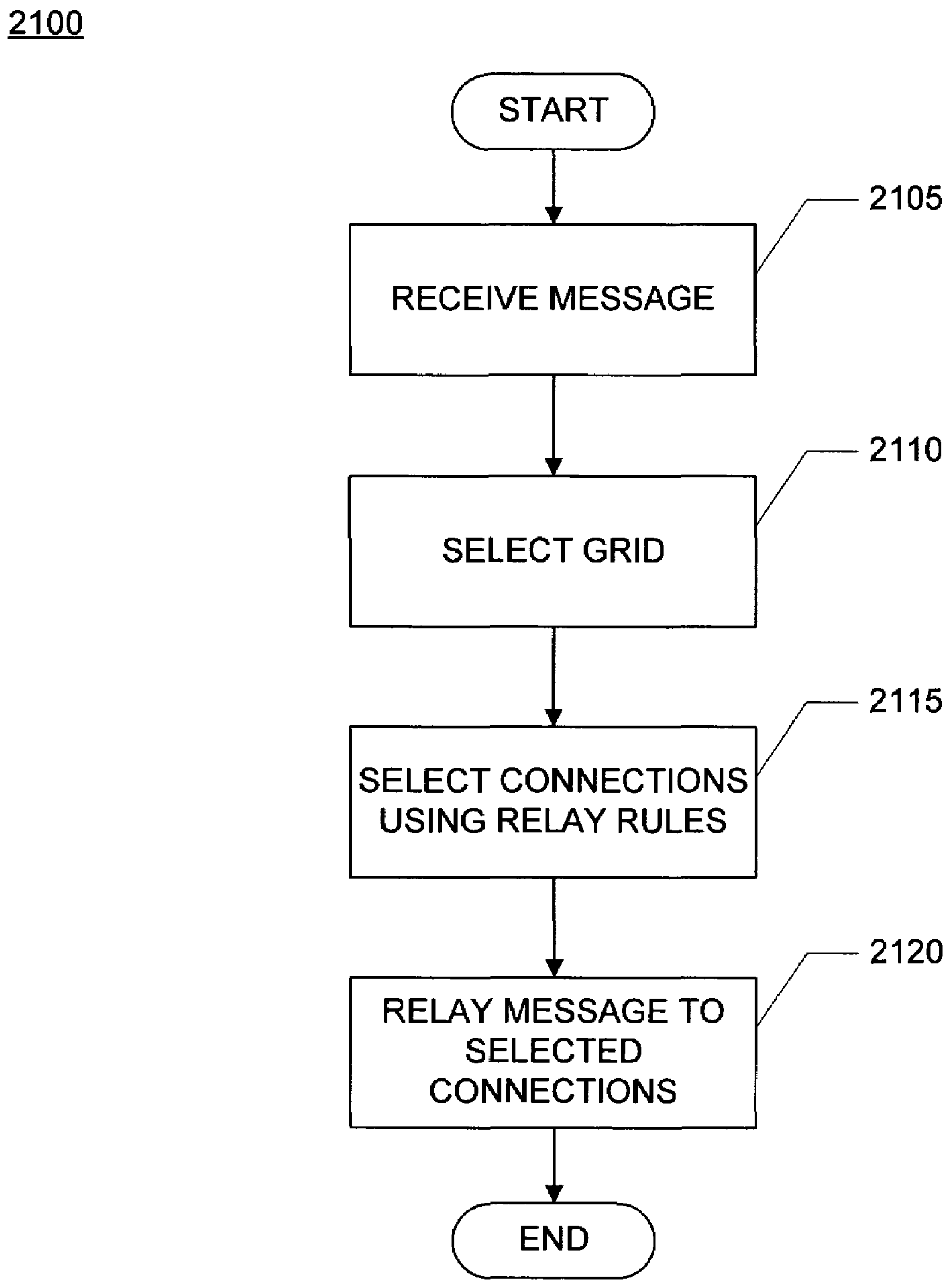

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

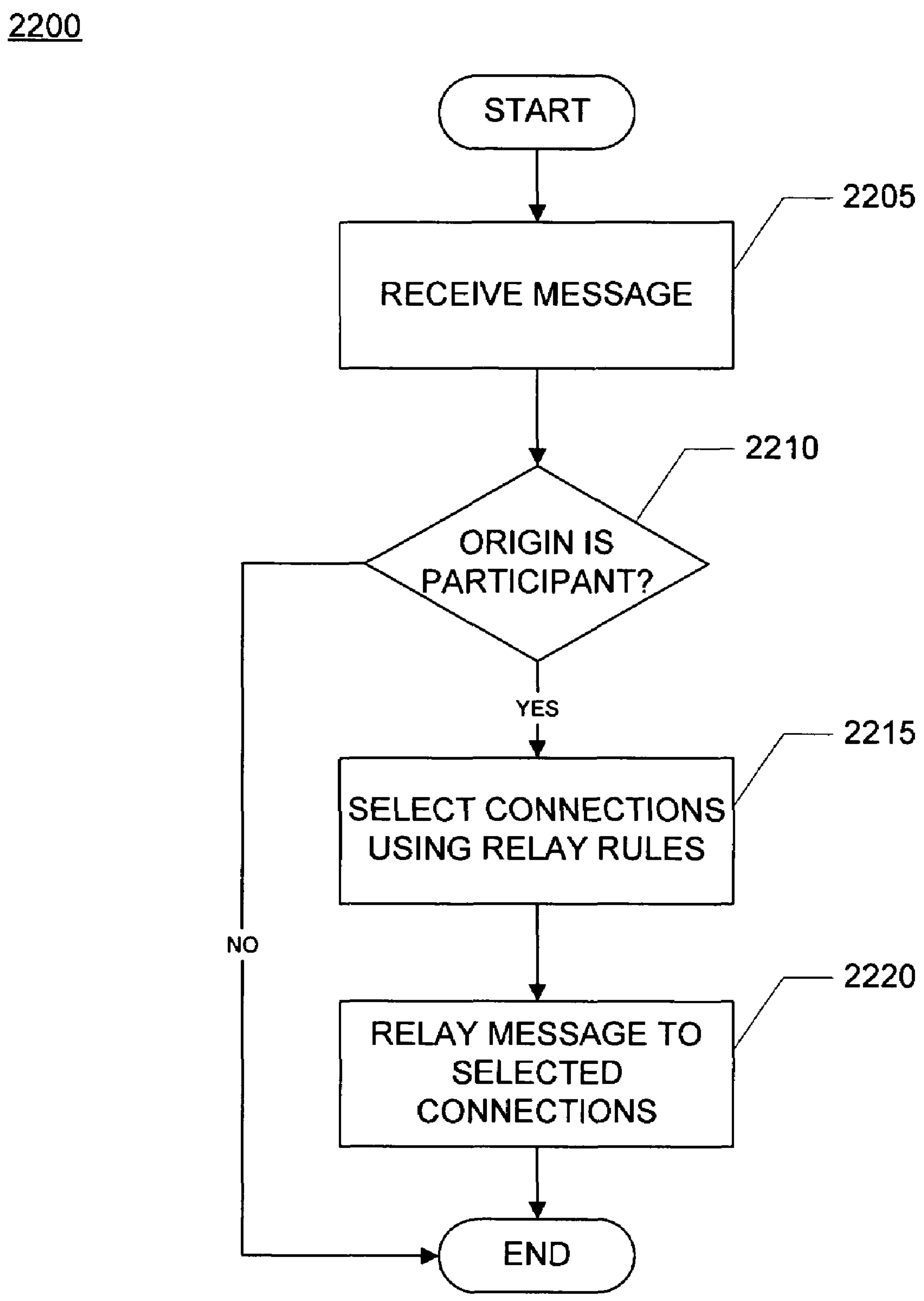

2200
START
2205
RECEIVE MESSAGE
2210
ORIGIN IS PARTICIPANT?
YES
NO
2215
SELECT CONNECTIONS USING RELAY RULES
2220
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 23

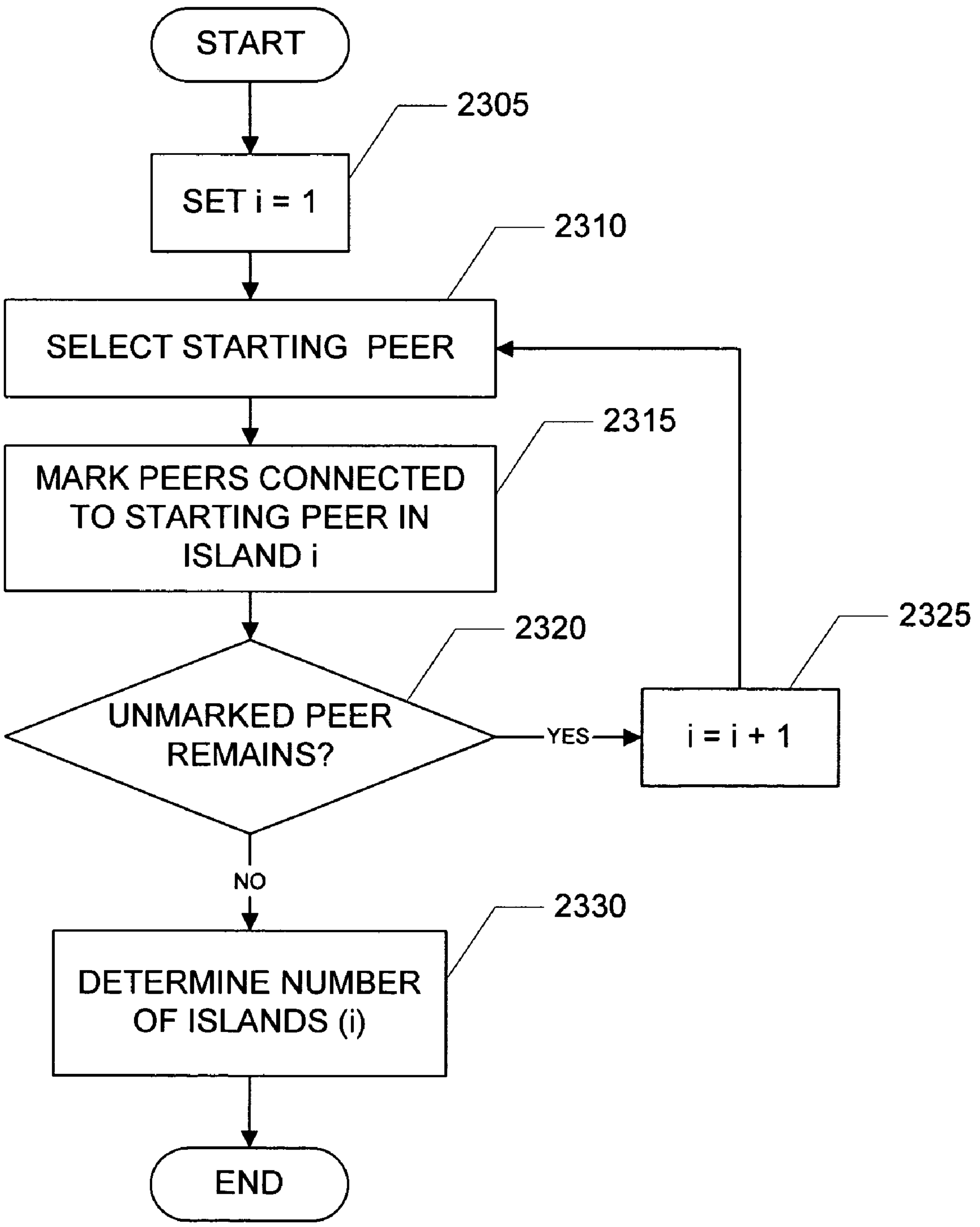

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

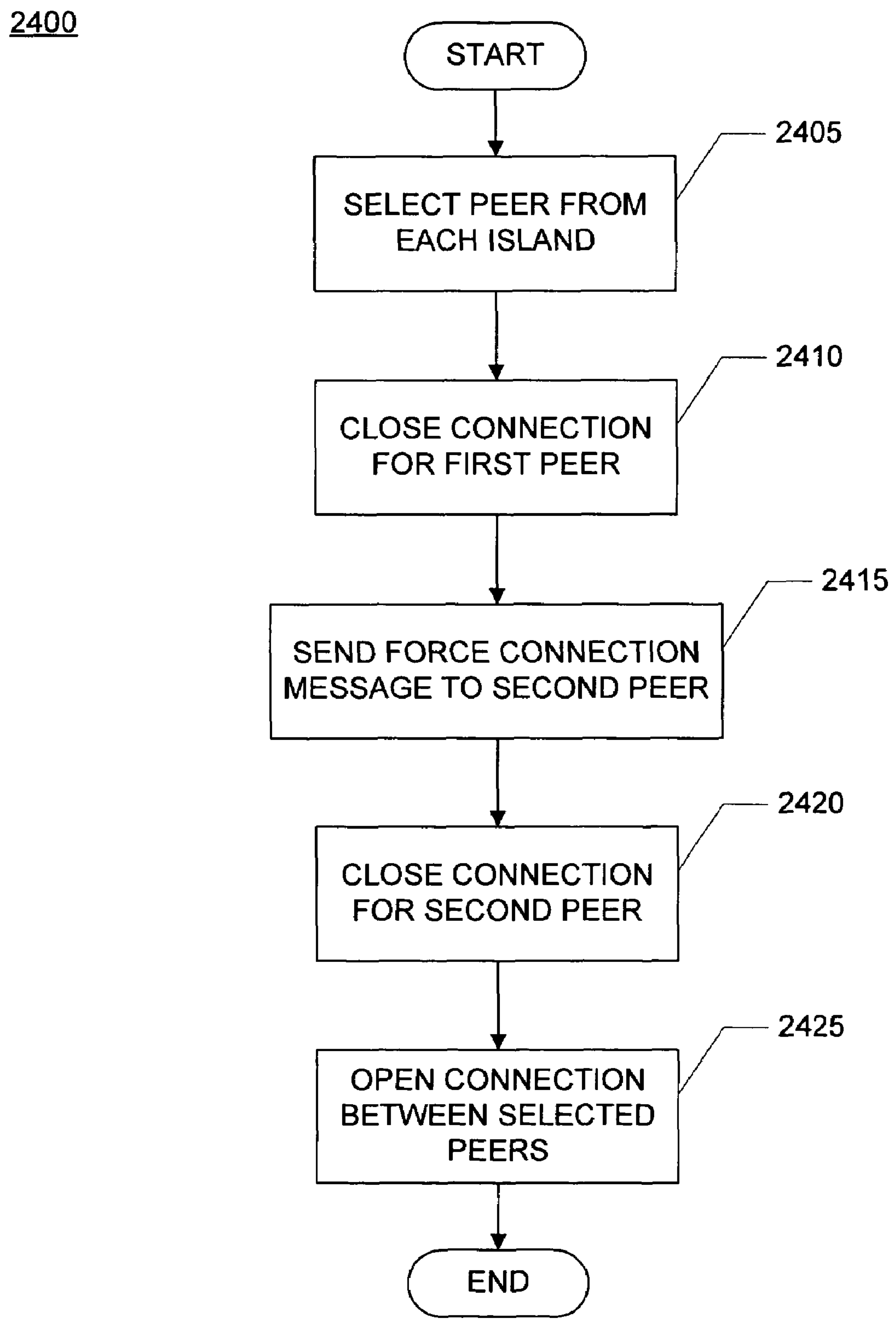

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

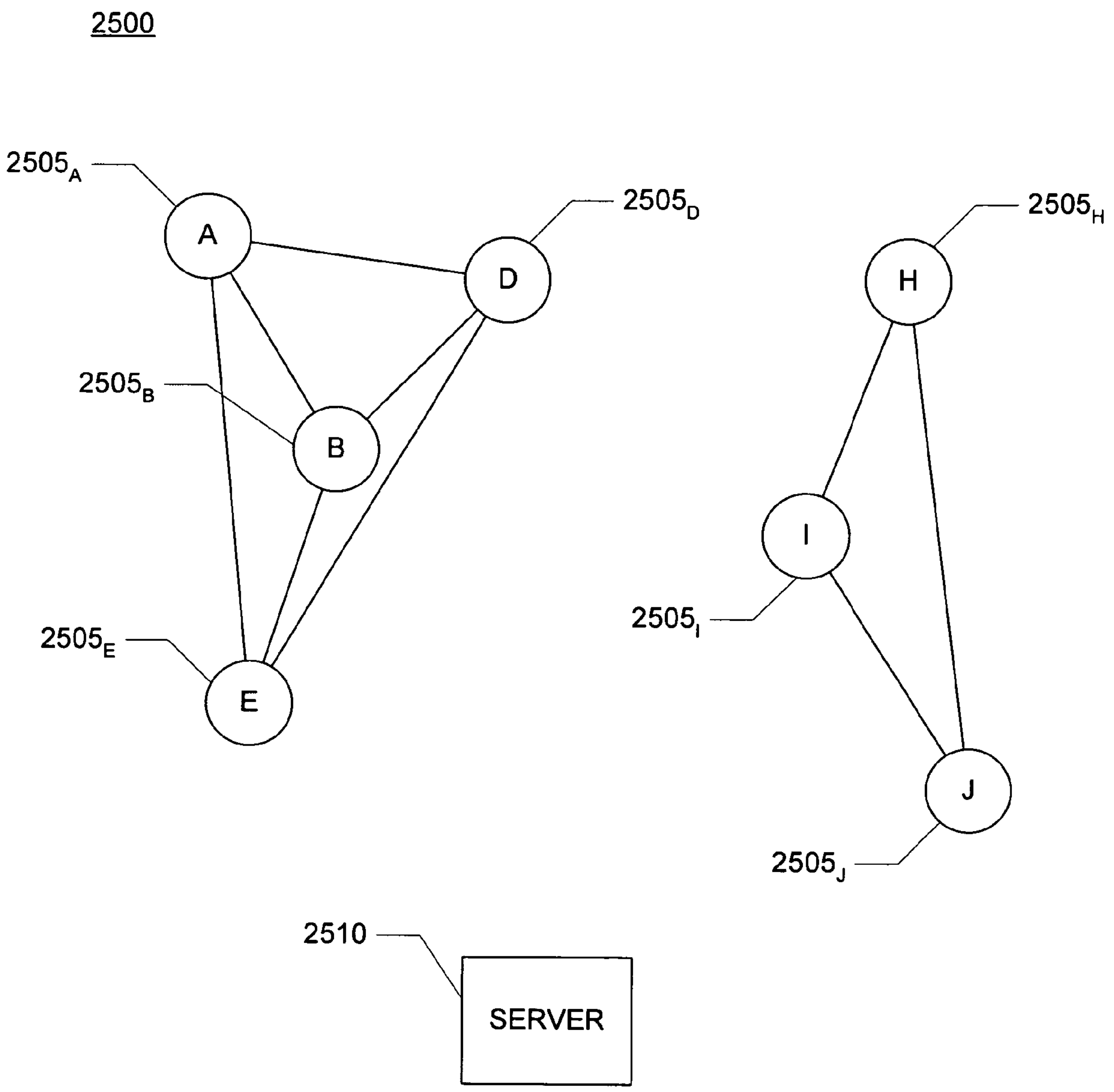
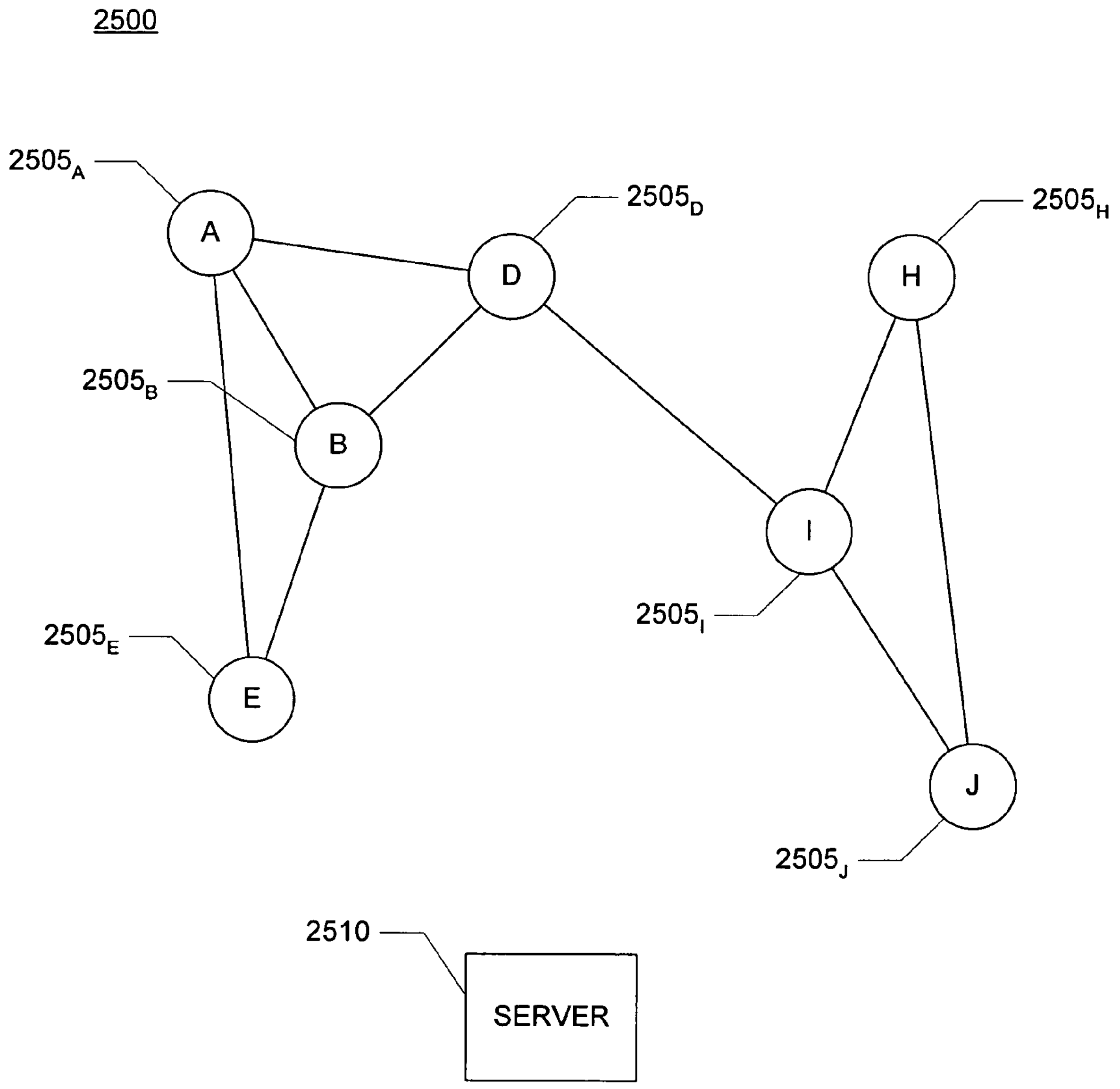

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 26



2500
2505_A
A
2505_D
D
2505_H
H
2505_B
B
I
2505_I
2505_E
E
J
2505_J
2510
SERVER

FIG. 27

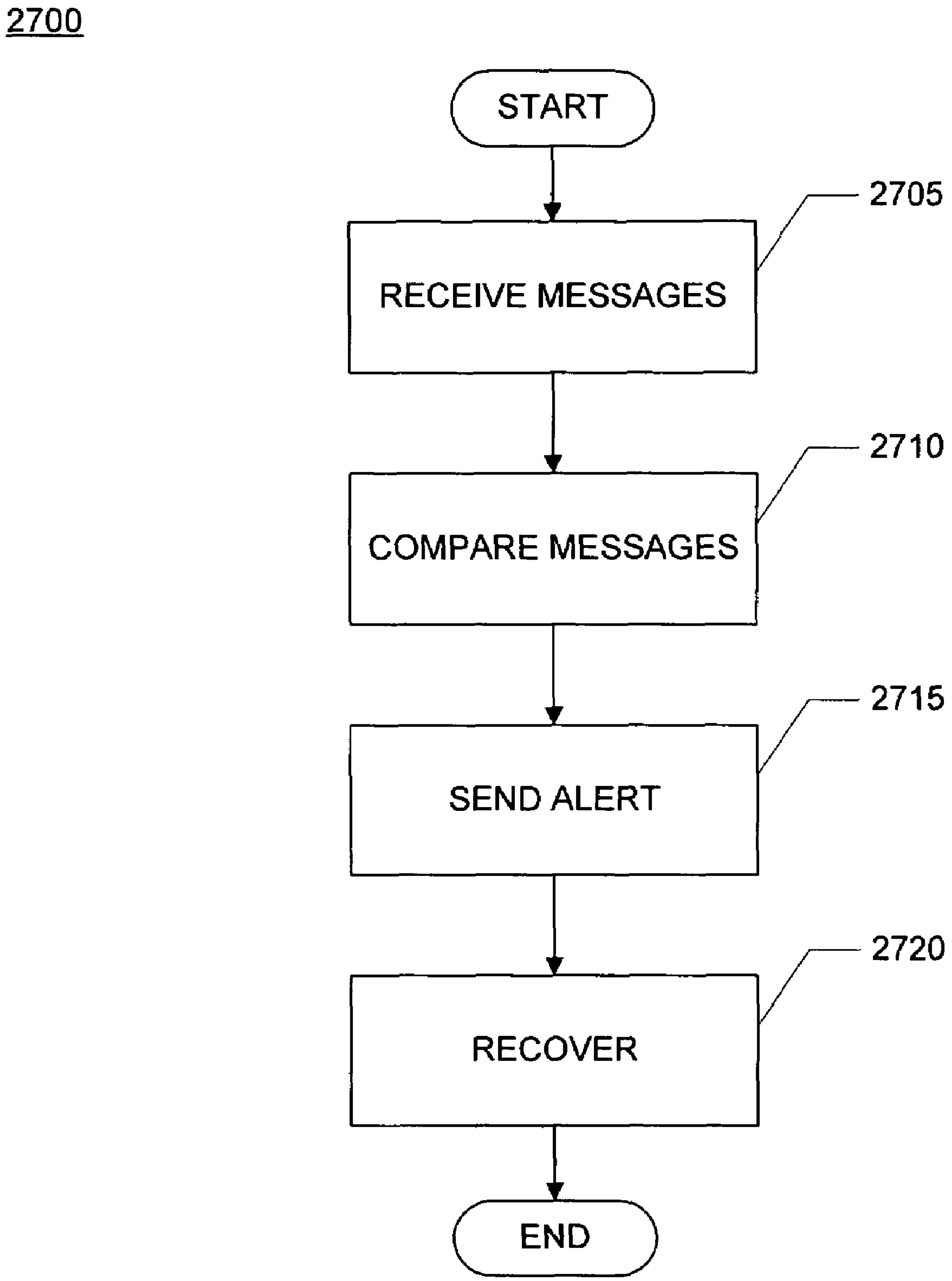

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

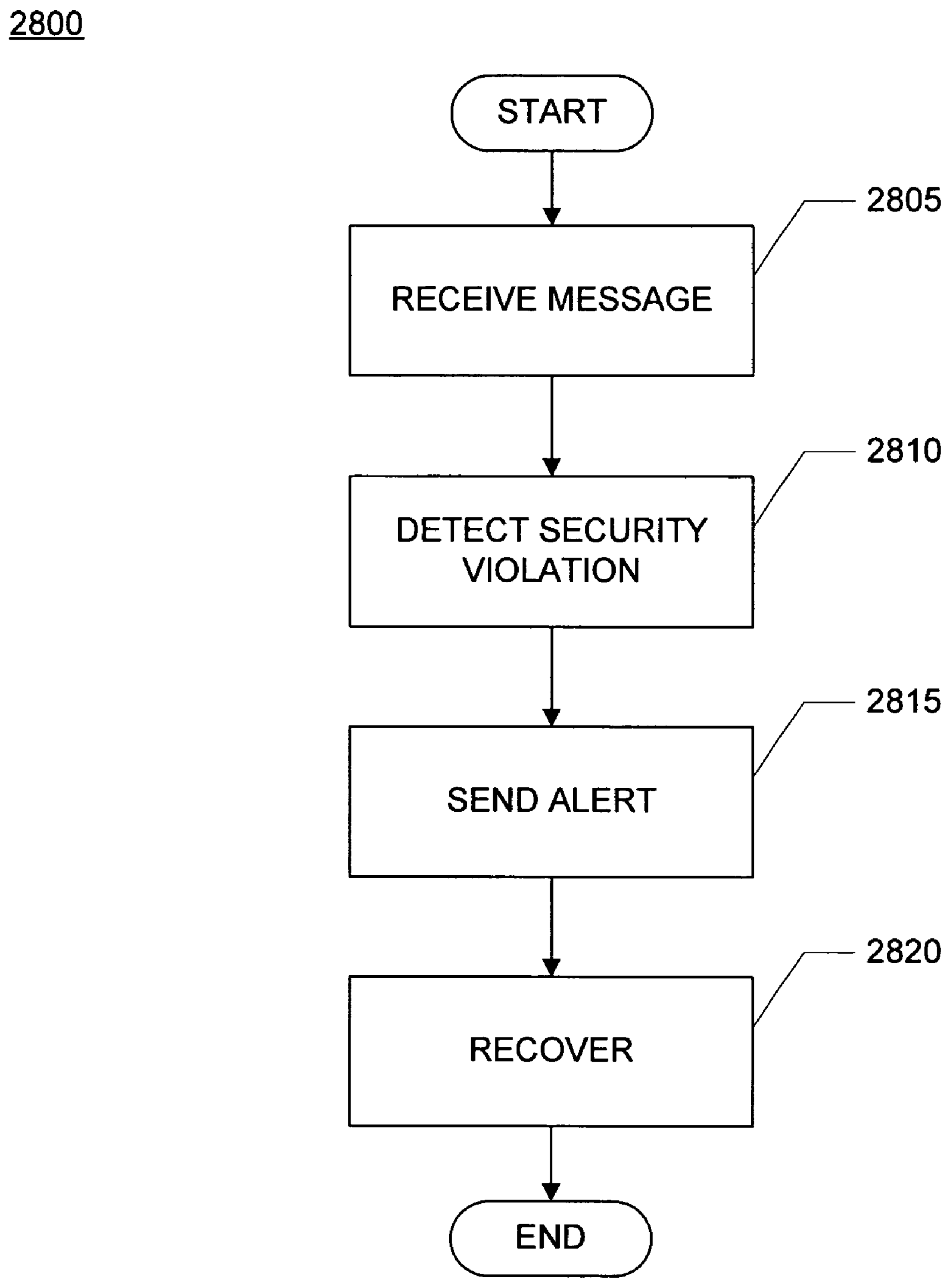

2800
START
RECEIVE MESSAGE
2805
DETECT SECURITY VIOLATION
2810
SEND ALERT
2815
RECOVER
2820
END

FIG. 29

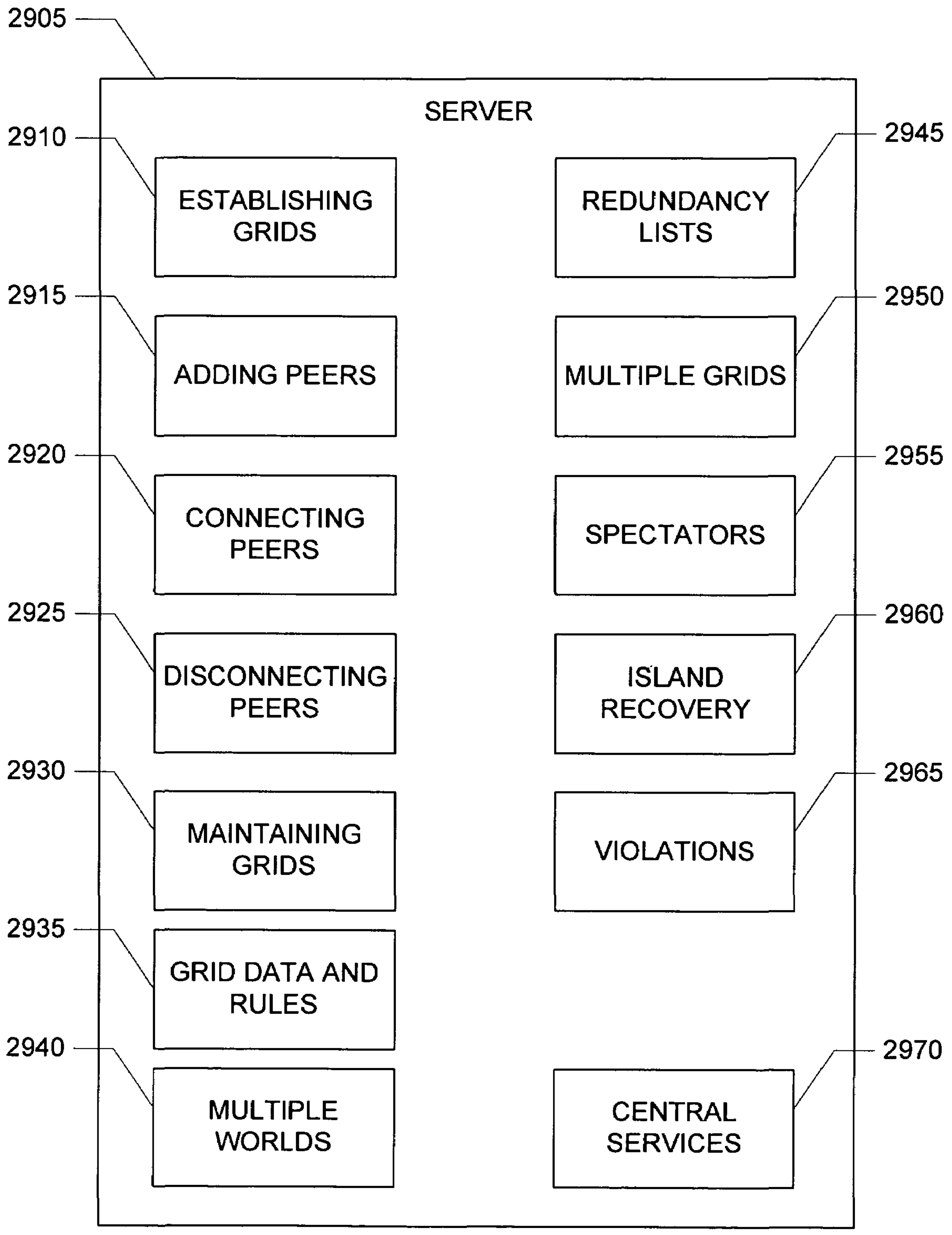

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

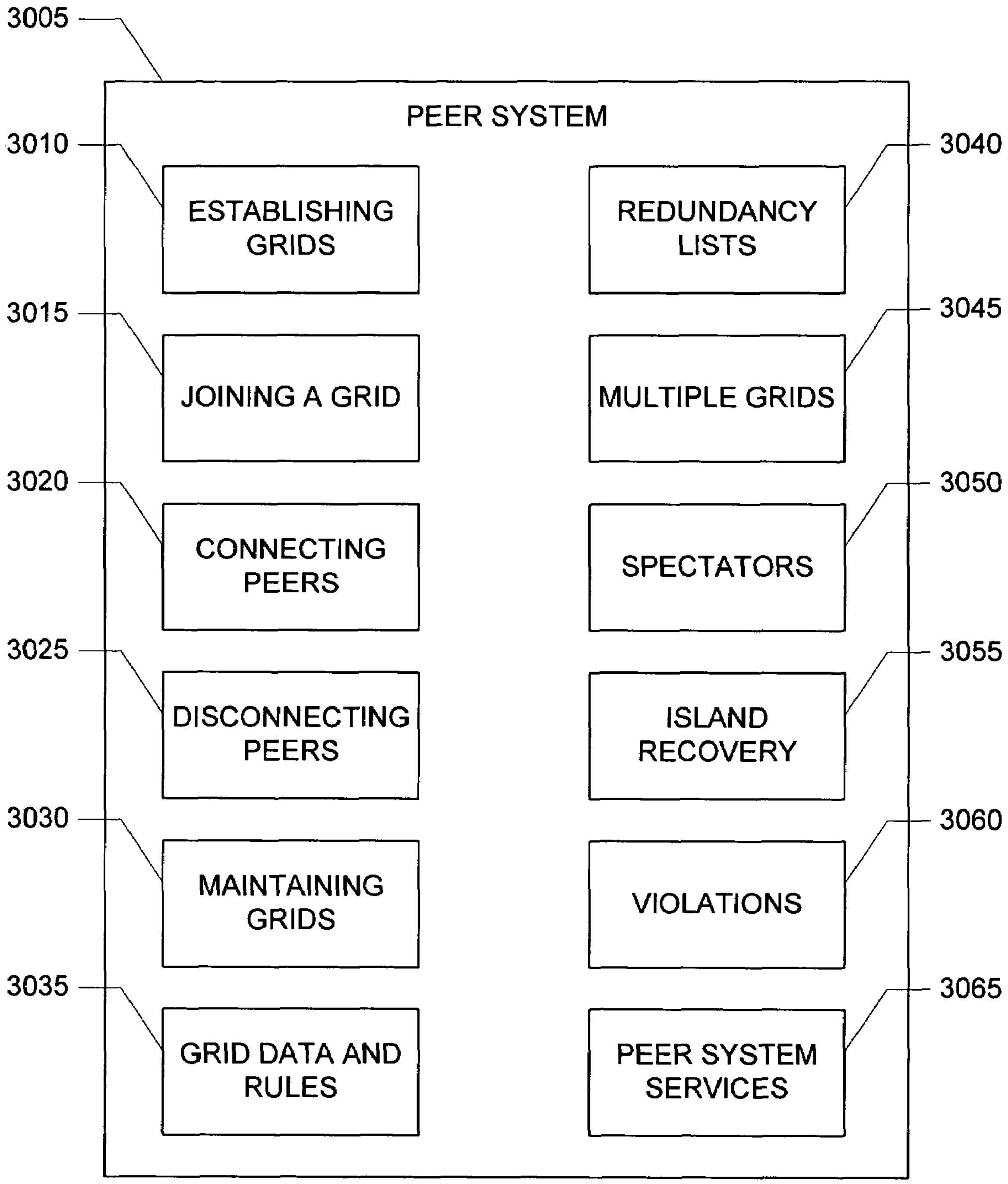

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

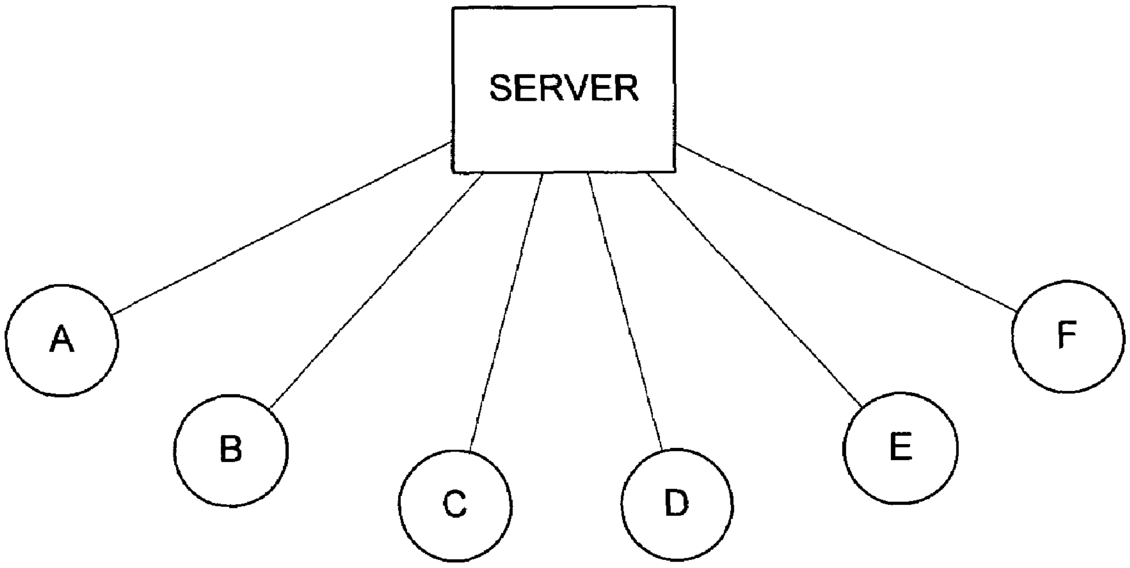
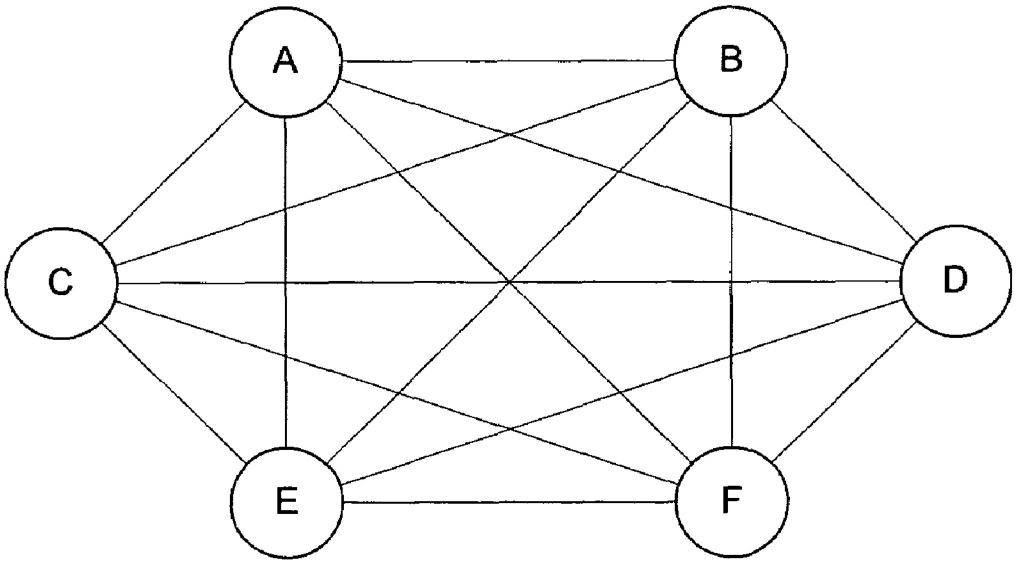


FIG. 31B



ISLAND RECOVERY IN A PEER-TO-PEER RELAY NETWORK

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

This application is related to the U.S. applications Ser. No. 10/700,798, filed on Nov. 3, 2003, Ser. No. 10/701,302, filed on Nov. 3, 2003, Ser. No. 10/701,014, filed on Nov. 3, 2003, Ser. No. 10/700,777, filed on Nov. 3, 2003, and Ser. No. 10/700,797, filed on Nov. 3, 2003.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N-1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of island recovery in a peer system in a peer-to-peer relay network includes: detecting the presence of two or more islands in a peer-to-peer relay network, wherein each island includes at least one peer system; joining two detected islands by connecting a peer system in a first island to a peer system in a second island; wherein peer systems in different islands are not connected.

In one implementation, a server in a peer-to-peer relay network includes: means for detecting the presence of two or more islands in a peer-to-peer relay network, wherein each island includes at least one peer system; means for joining two detected islands by connecting a peer system in a first island to a peer system in a second island; wherein peer systems in different islands are not connected.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message, in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent applications Ser. Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed 31 Jul. 2002 now U.S. Pat. No. 7,421,471), and 10/359,359 ("Multi-User Application Programming Interface"), filed 4 Feb. 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system $\mathbf{105}_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying-peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection 1, peer B is connected to connection 2, and peer G is connected to connection 3. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed.

The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the, peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins, peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive

responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel

spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B. D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method, executed by a processor, of island recovery in a peer system in a peer-to-peer relay network, comprising:

tracking all connections among peer systems in the peer-to-peer relay network;

detecting a total number of islands in the peer-to-peer relay network by sequentially detecting and marking every peer system in the peer-to-peer relay network,

wherein the detecting step detects whether an unmarked peer system has a peer-to-peer connection with any of existing islands, and when the unmarked peer system has no peer-to-peer connection with any of the existing islands, the peer system is marked with an identifier indicating a new island and is treated as a starting peer of the new island, and

wherein each island includes at least one peer system; and

joining two detected islands by connecting a peer system in a first island to a peer system in a second island through a peer-to-peer connection,

wherein the peer system in a first island sends to the peer system in a second island a force connection message that, if the peer system in the second island does not have an available peer-to-peer connection, causes the peer system in the second island to select an existing peer-to-peer connection and close the selected existing peer-to-peer connection, the peer system in the first island and the peer system in the second island being randomly selected, and

wherein peer systems in the first island are not connected with peer systems in the second island through a peer-to-peer connection before joining the first island and the second island.

2. The method of claim **1**, wherein:

detecting step includes:

setting an island counter to 1;

(a) selecting a first unmarked peer system as a starting peer system,

(b) marking peer systems connected to the first unmarked peer system according to the value of said island counter,

(c) determining if a second unmarked peer system in said peer-to-peer relay network remains,

(d) if the second unmarked peer system remains, incrementing said island counter;

repeating steps (a) through (d) until each the peer system in said peer-to-peer relay network has been marked; and

determining the number of islands present using the current value of said island counter.

3. The method of claim **1**, wherein:

joining two detected islands includes:

selecting a first peer system from a first detected island and a second peer system from a second detected island; and

causing the first peer system in the first detected island to open a connection to the second peer system in the second detected island.

4. The method of claim **1**, wherein:

each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and

each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

5. The method of claim **1**, wherein: the data relayed by peer systems is update data for a network environment.

6. The method of claim **1**, wherein: the data relayed by peer systems is update data for an online game.

7. The method of claim **1**, wherein: at least one peer system is a network-enabled game console.

8. The method of claim **1**, wherein: at least two peer systems are connected through the Internet.

9. A server computer in a peer-to-peer relay network, comprising:

means for tracking all connections among peer systems in the peer-to-peer relay network;

means for detecting a total number of islands in a peer-to-peer relay network by sequentially detecting and marking every peer system in the peer-to-peer relay network,

wherein the detecting means detects whether an unmarked peer system has a peer-to-peer connection with any of existing islands, and when the unmarked peer system has no peer-to-peer connection with any of the existing islands, the peer system is marked with an identifier indicating a new island and is treated as a starting peer of the new island, and

wherein each island includes at least one peer system; and

means for joining two detected islands by connecting a peer system in a first island to a peer system in a second island through a peer-to-peer connection,

wherein the peer system in a first island sends to the peer system in a second island a force connection message that, if the peer system in the second island does not have an available peer-to-peer connection, causes the peer system in the second island to select an existing peer-to-peer connection and close the selected existing peer-to-peer connection, the peer system in the first island and the peer system in the second island being randomly selected, and

wherein peer systems in the first island are not connected with peer systems in the second island through a peer-to-peer connection before joining the first island and the second island.

10. The server of claim **9**, wherein:

said means for detecting a total number of islands includes:

means for setting an island counter to 1, wherein the means for setting includes:
(a) selecting a first unmarked peer system as a starting peer system,
(b) marking peer systems connected to the first unmarked peer system according to the value of said island counter,
(c) determining if a second unmarked peer system in said peer-to-peer relay network remains,
(d) if a second unmarked peer system remains, incrementing said island counter;
means for repeating steps (a) through (d) until each peer system in said peer-to-peer relay network has been marked; and
means for determining the number of islands present using the current value of said island counter.

11. The server of claim **9**, wherein:
said means for joining two detected islands includes:
means for selecting a first peer system from a first detected island and a second peer system from a second detected island; and
means for causing the first peer system in the first detected island to open a connection to the second peer system in the second detected island.

12. The server of claim **9**, wherein:
each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

13. A computer program, stored on a storage medium, for use in island recovery in a peer-to-peer relay network, the program comprising executable instructions that cause a computer to:
track all connections among peer systems in the peer-to-peer relay network;
detect a total number of islands in the peer-to-peer relay network by sequentially detecting and marking every peer system in the peer-to-peer relay network,
wherein the detecting step detects whether an unmarked peer system has a peer-to-peer connection with any of existing islands, and when the unmarked peer system has no peer-to-peer connection with any of the existing islands, the peer system is marked with an identifier indicating a new island and is treated as a starting peer of the new island, and
wherein each island includes at least one peer system; and

join two detected islands by connecting a peer system in a first island to a peer system in a second island through a peer-to-peer connection,
wherein the peer system in a first island sends to the peer system in a second island a force connection message that, if the peer system in the second island does not have an available peer-to-peer connection, causes the peer system in the second island to select an existing peer-to-peer connection and close the selected existing peer-to-peer connection, the peer system in the first island and the peer system in the second island being randomly selected, and
wherein peer systems in the first island are not connected with peer systems in the second island through a peer-to-peer connection before joining the first island and the second island.

14. The computer program of claim **13**, wherein:
the detecting step includes:
setting an island counter to 1;
(a) selecting a first unmarked peer system as a starting peer system,
(b) marking peer systems connected to the first unmarked peer system according to the value of said island counter,
(c) determining if a second unmarked peer system in said peer-to-peer relay network remains,
(d) if a second unmarked peer system remains, incrementing said island counter;
repeating steps (a) through (d) until each peer system in said peer-to-peer relay network has been marked; and
determining the number of islands present using the current value of said island counter.

15. The computer program of claim **13**, wherein:
joining two detected islands includes:
selecting a first peer system from a first detected island and a second peer system from a second detected island; and
causing the first peer system in the first detected island to open a connection to the second peer system in the second detected island.

16. The computer program of claim **13**, wherein:
each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and
each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 7,596,633 B2	Page 1 of 1
APPLICATION NO.	: 10/701298	
DATED	: September 29, 2009	
INVENTOR(S)	: Mai et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1142 days.

Signed and Sealed this

Twenty-eighth Day of September, 2010

David J. Kappos

David J. Kappos

Director of the United States Patent and Trademark Office

EXHIBIT 84

US007610402B2

(12) **United States Patent**
Van Datta

(10) **Patent No.: US 7,610,402 B2**
(45) **Date of Patent: Oct. 27, 2009**

(54) **SPECTATORS IN A PEER-TO-PEER RELAY NETWORK**

(75) Inventor: **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Foster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1478 days.

(21) Appl. No.: **10/700,777**

(22) Filed: **Nov. 3, 2003**

(65) **Prior Publication Data**

US 2005/0086287 A1 Apr. 21, 2005

Related U.S. Application Data

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 13/00 (2006.01)

(52) **U.S. Cl.** **709/238**; 709/220; 709/226; 709/250

(58) **Field of Classification Search** 709/220, 709/222, 223, 225, 227, 228, 230, 238, 250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,398,388	B2 *	7/2008	Xu et al.	713/163
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0119821	A1	8/2002	Sen et al.	
2002/0143855	A1 *	10/2002	Traversat et al.	709/202
2002/0184310	A1	12/2002	Traversat et al.	
2003/0055892	A1	3/2003	Huitema et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999

(Continued)

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

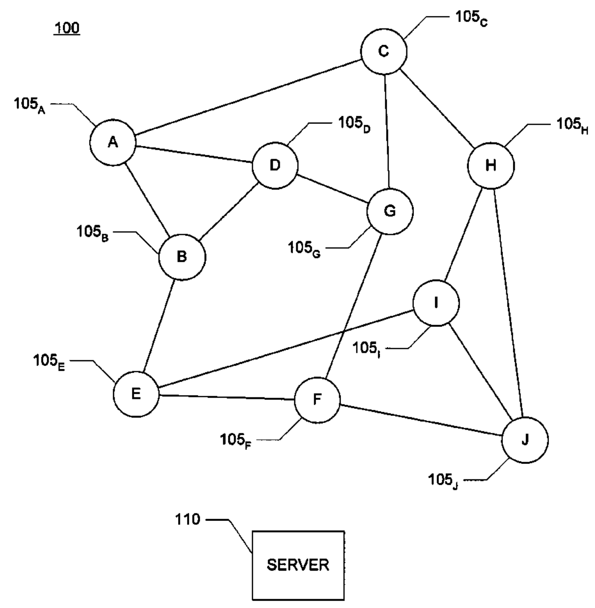
(Continued)

Primary Examiner—Viet Vu
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system in said peer-to-peer relay network is connected to a number of other peer systems in said peer-to-peer relay network that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, each peer system in said peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules, each peer system is a participant or a spectator, at least one peer system is a participant, at least one peer system is a spectator, a participant is configured to generate data to be relayed in said peer-to-peer relay network, and a spectator is configured to relay data generated by a participant.

17 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

2003/0182428	A1 *	9/2003	Li et al.	709/227
2004/0162871	A1 *	8/2004	Pabla et al.	709/201

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E ED—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y ED—Association for Computing Machinery: "Simple and Fault-Tolerant Key Agreement by Dynamic Collaborative Groups" Proceedings of the 7th ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. CONF. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

* cited by examiner

FIG. 1

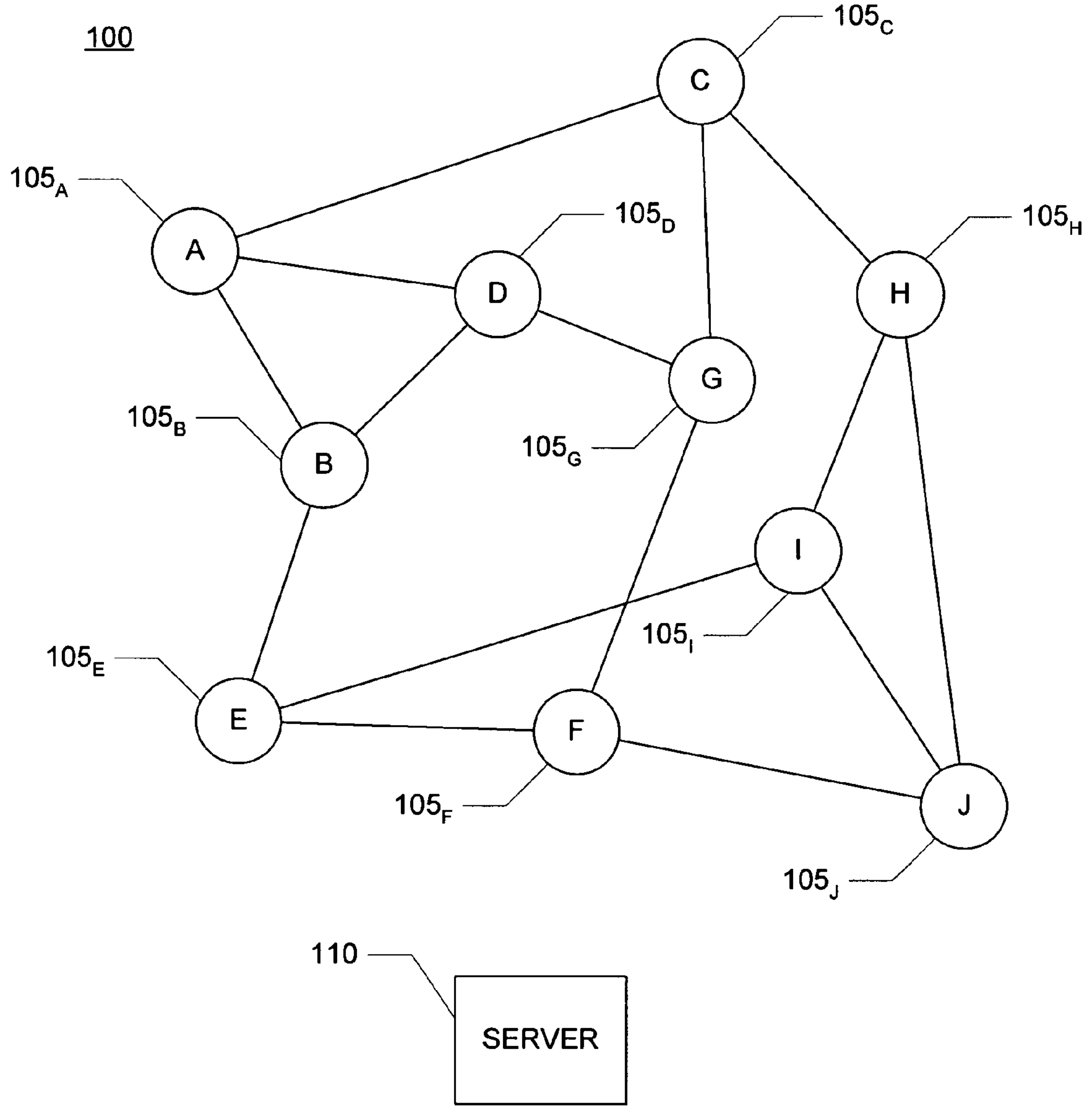

100
105C
C
105A
A
105D
D
105H
H
G
105B
B
105G
I
105I
105E
E
F
105F
J
105J
110
SERVER

FIG. 2

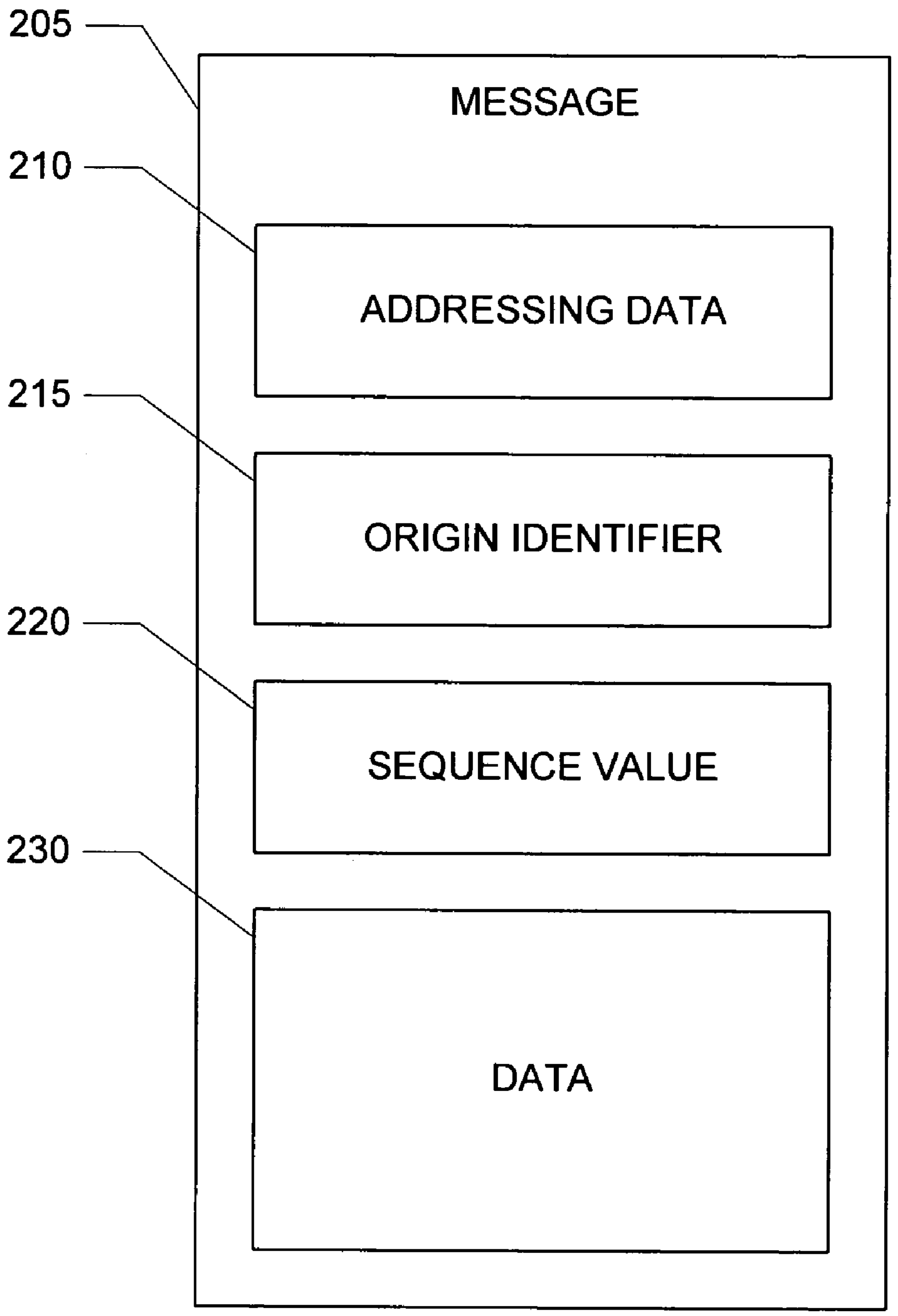

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

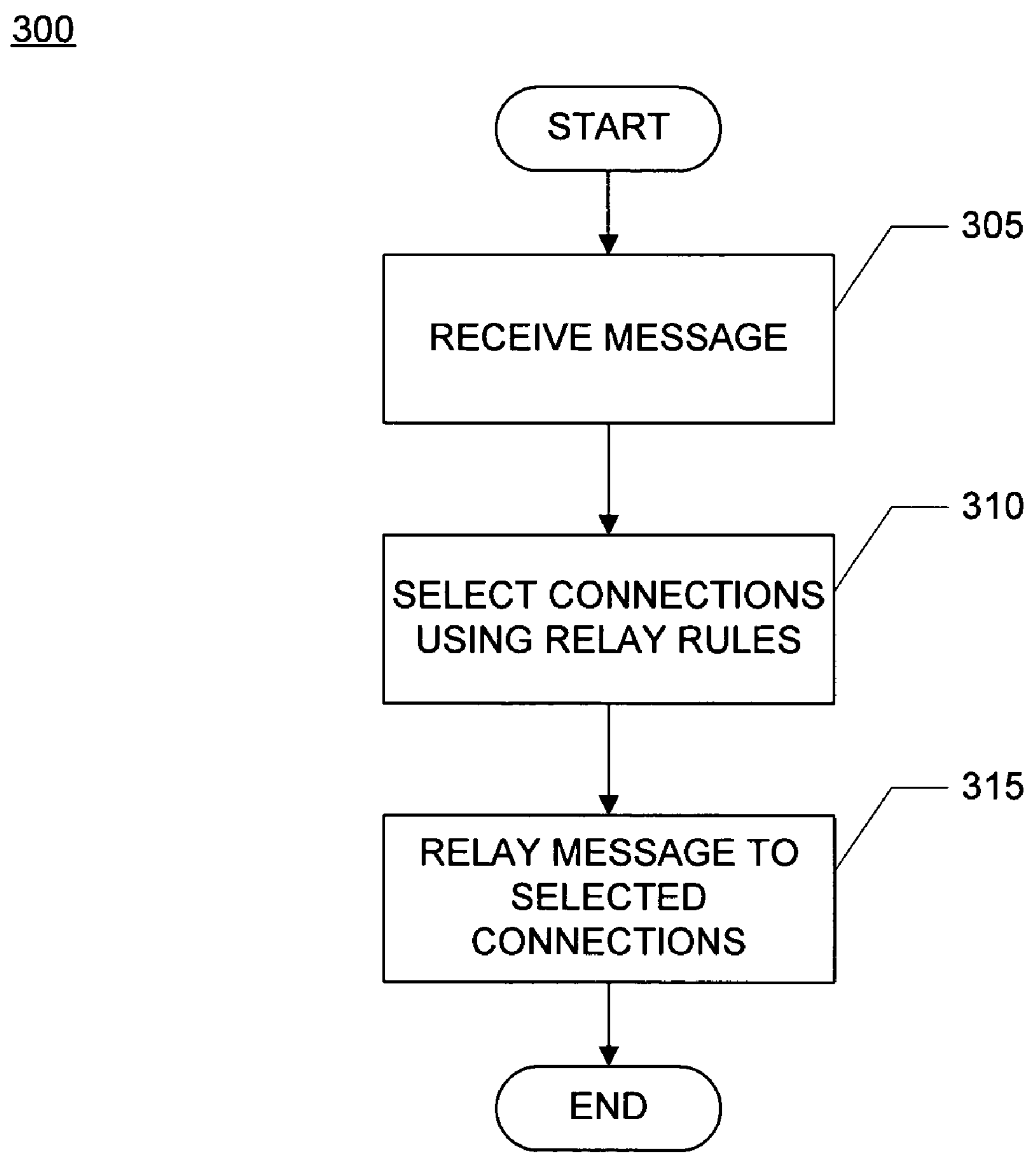

300
START
305
RECEIVE MESSAGE
310
SELECT CONNECTIONS USING RELAY RULES
315
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 4

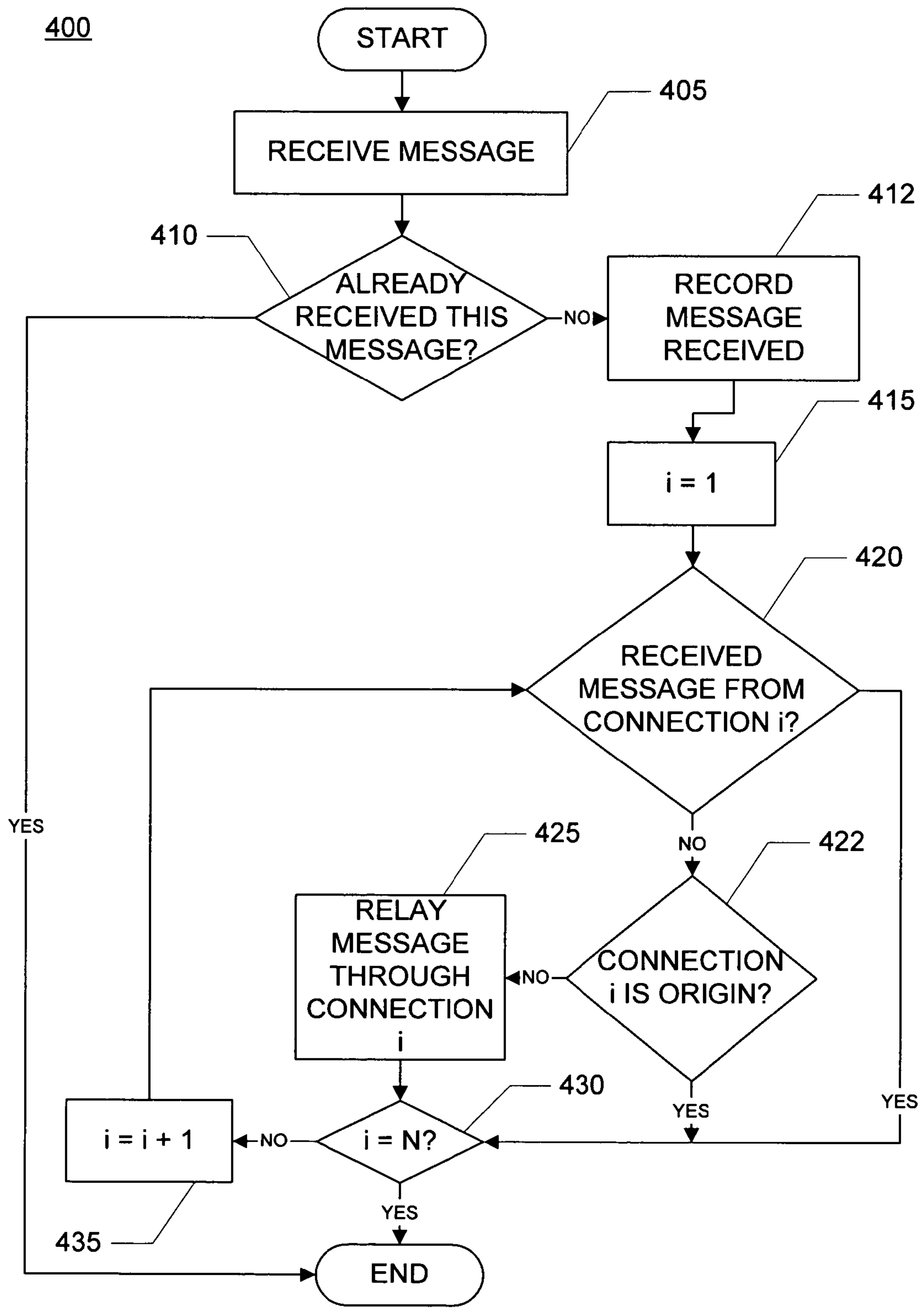

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
412
RECORD MESSAGE RECEIVED
415
i = 1
420
RECEIVED MESSAGE FROM CONNECTION i?
NO
422
CONNECTION i IS ORIGIN?
NO
425
RELAY MESSAGE THROUGH CONNECTION i
YES
YES
430
i = N?
NO
i = i + 1
435
YES
YES
END

FIG. 5

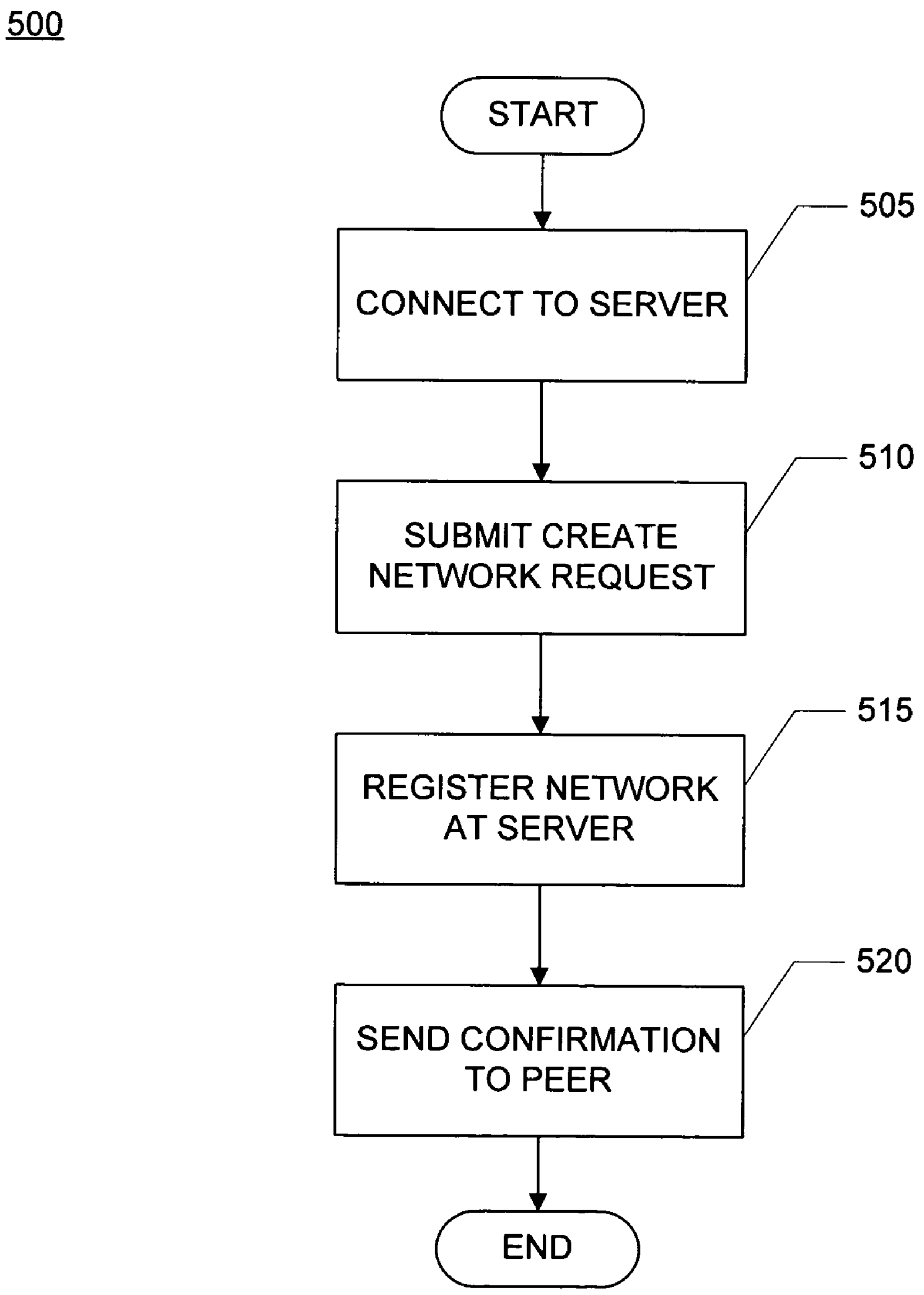

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

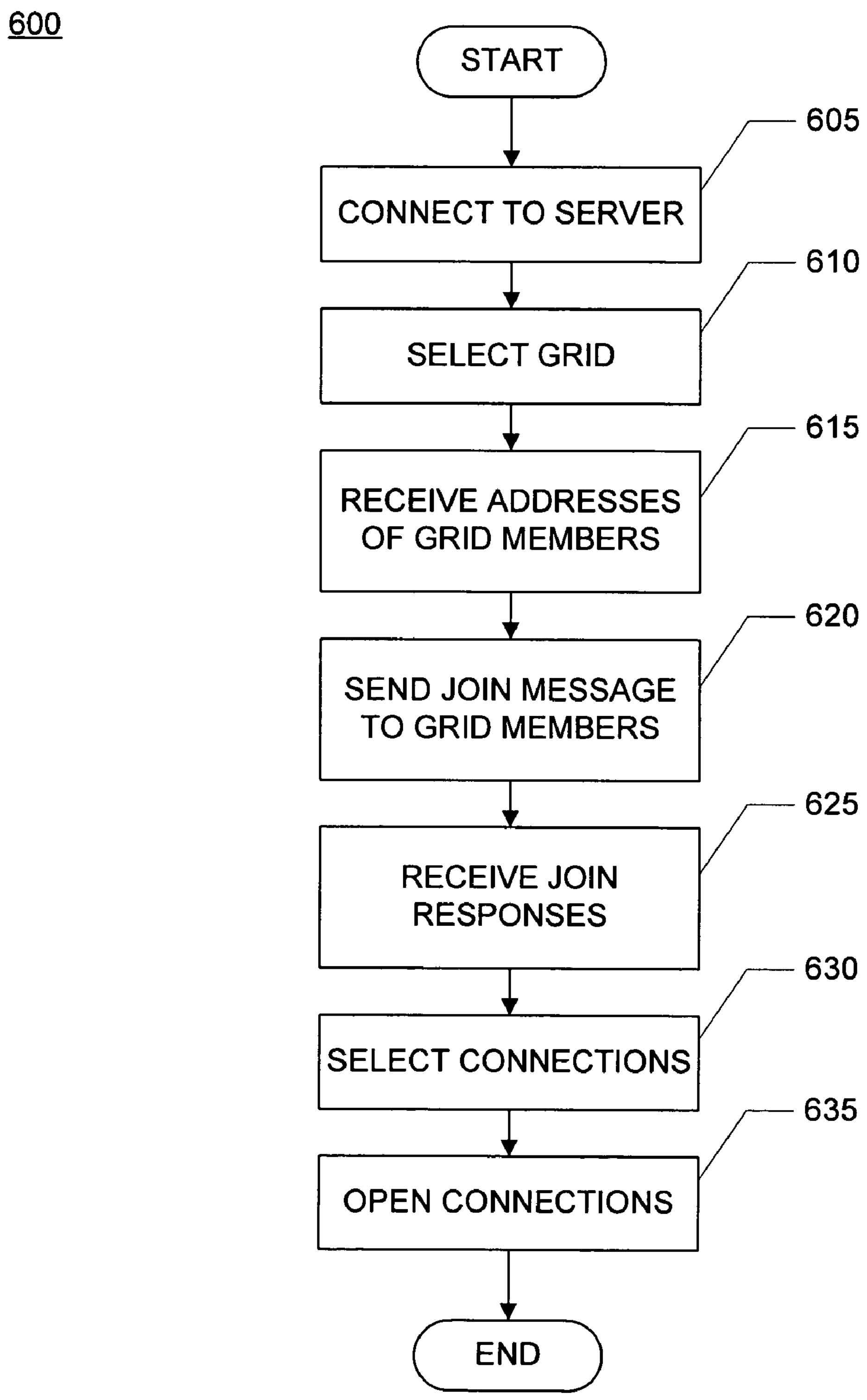

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

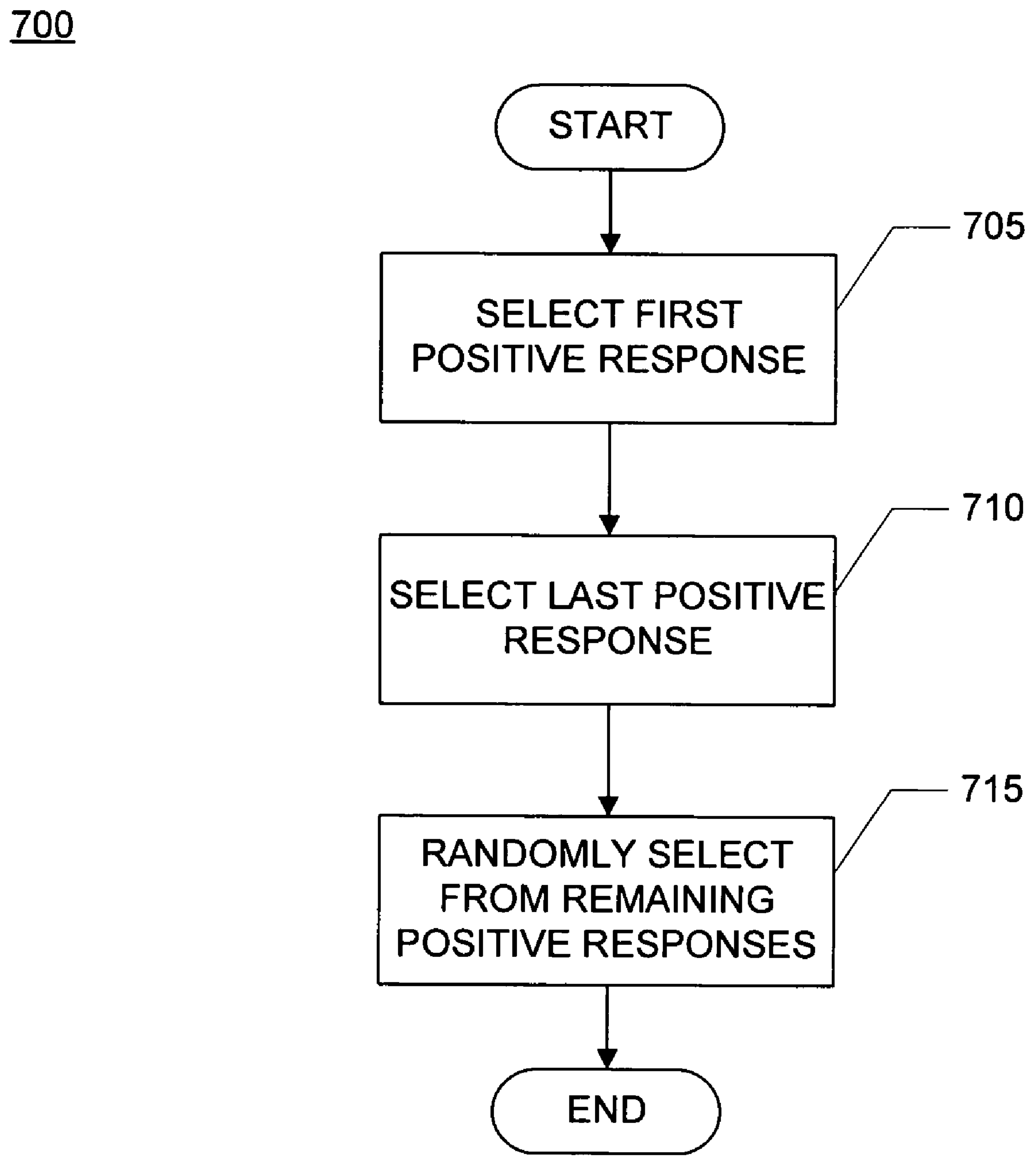

700
START
SELECT FIRST POSITIVE RESPONSE
705
SELECT LAST POSITIVE RESPONSE
710
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
715
END

FIG. 8

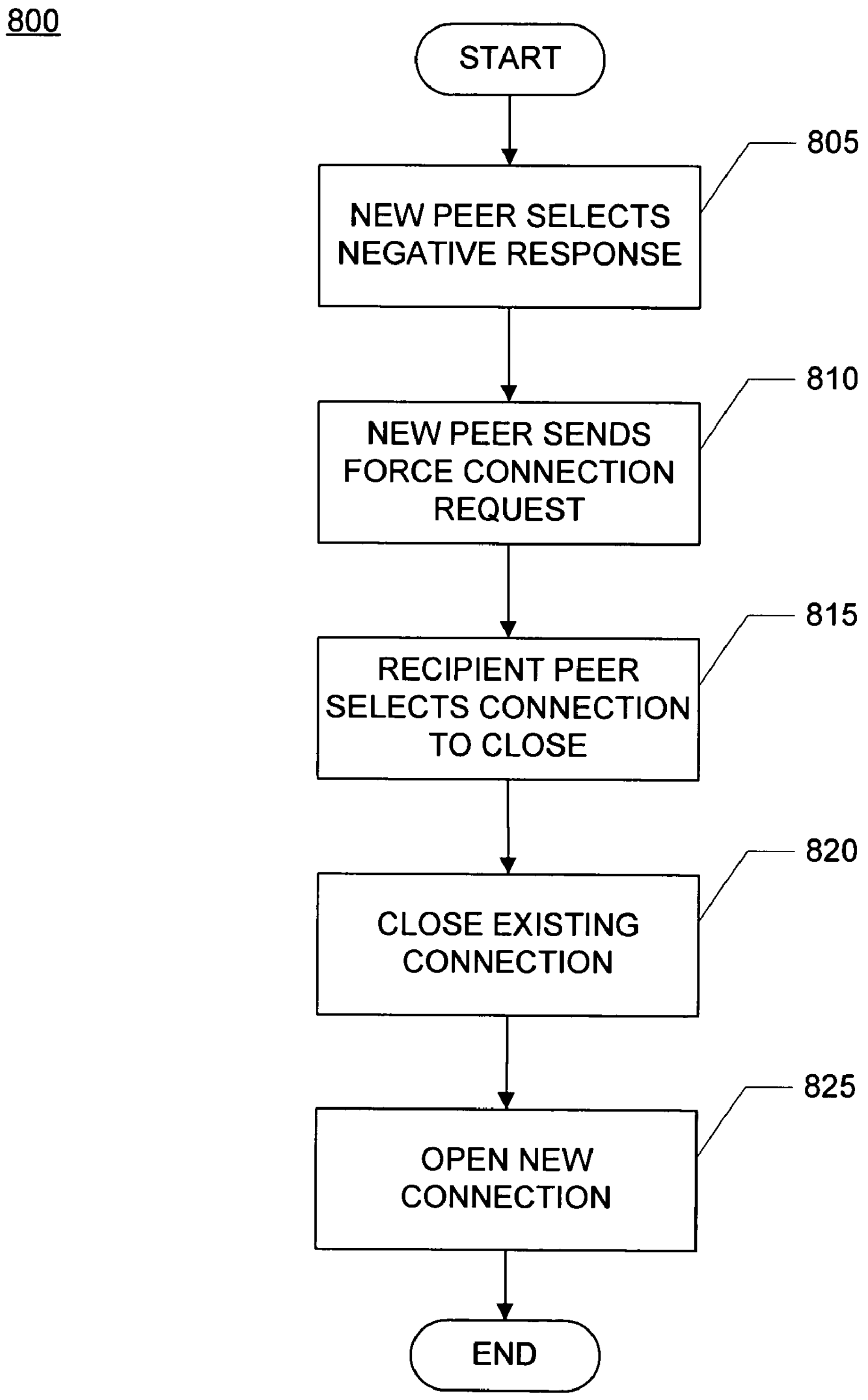

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

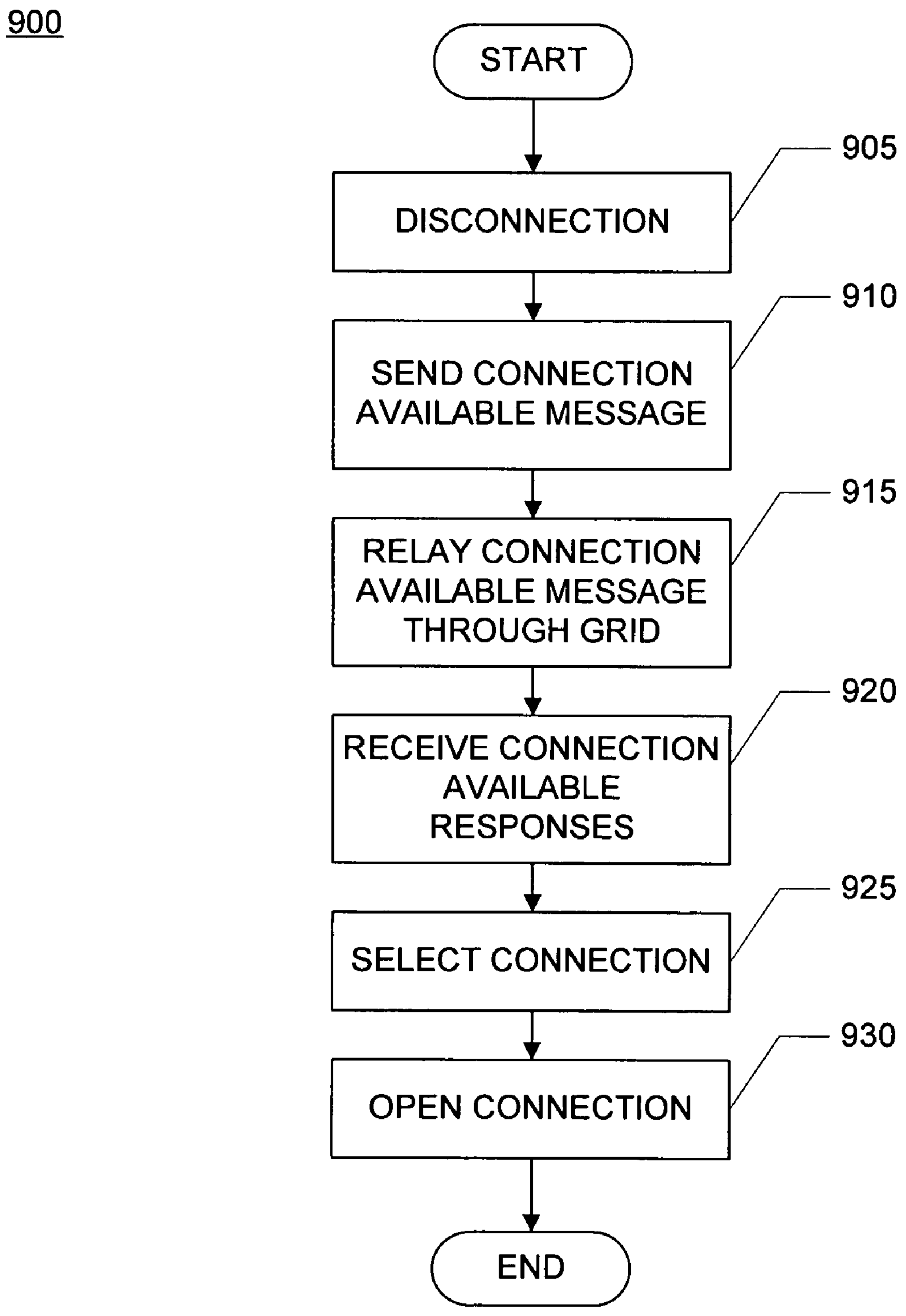

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

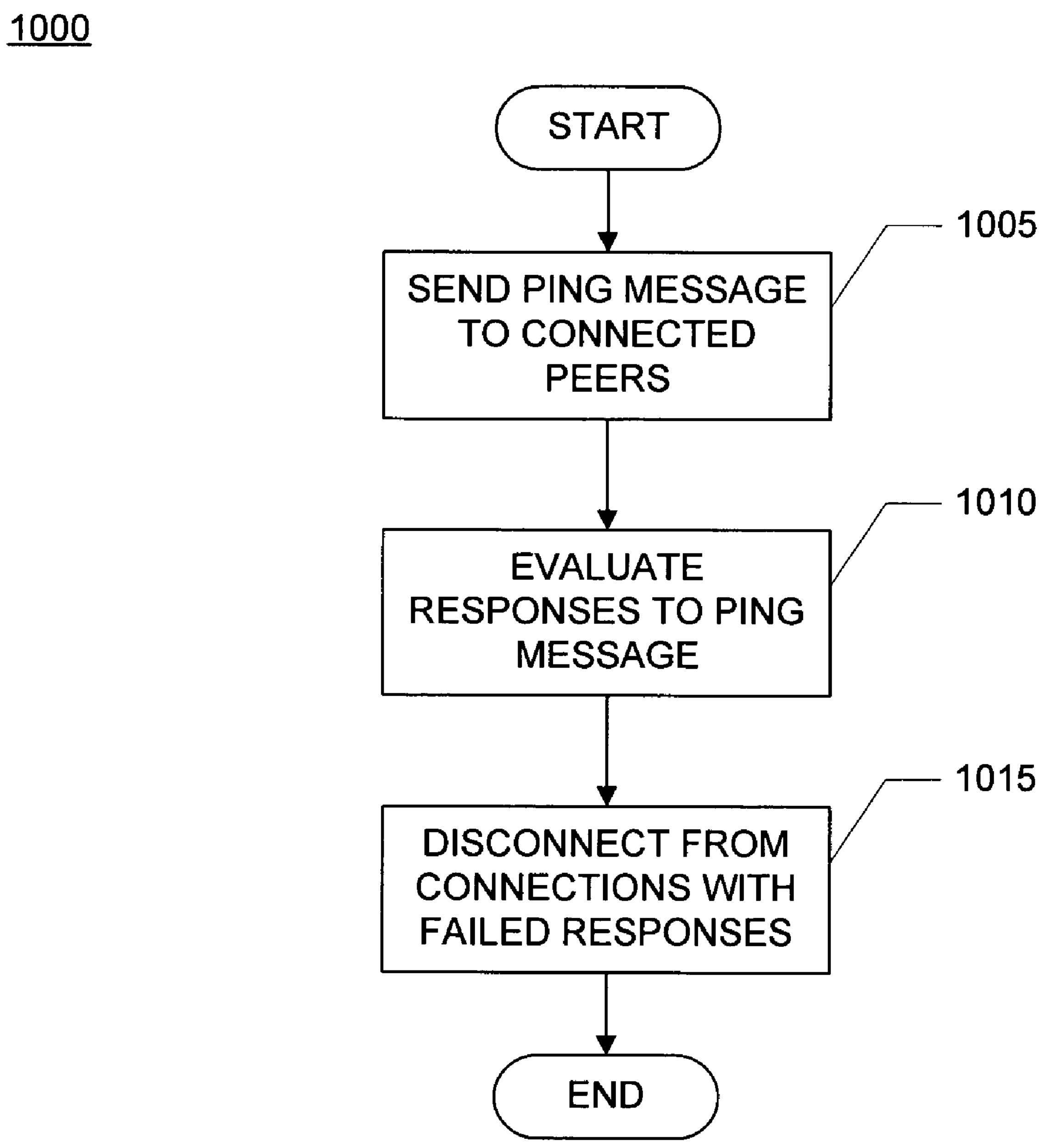

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

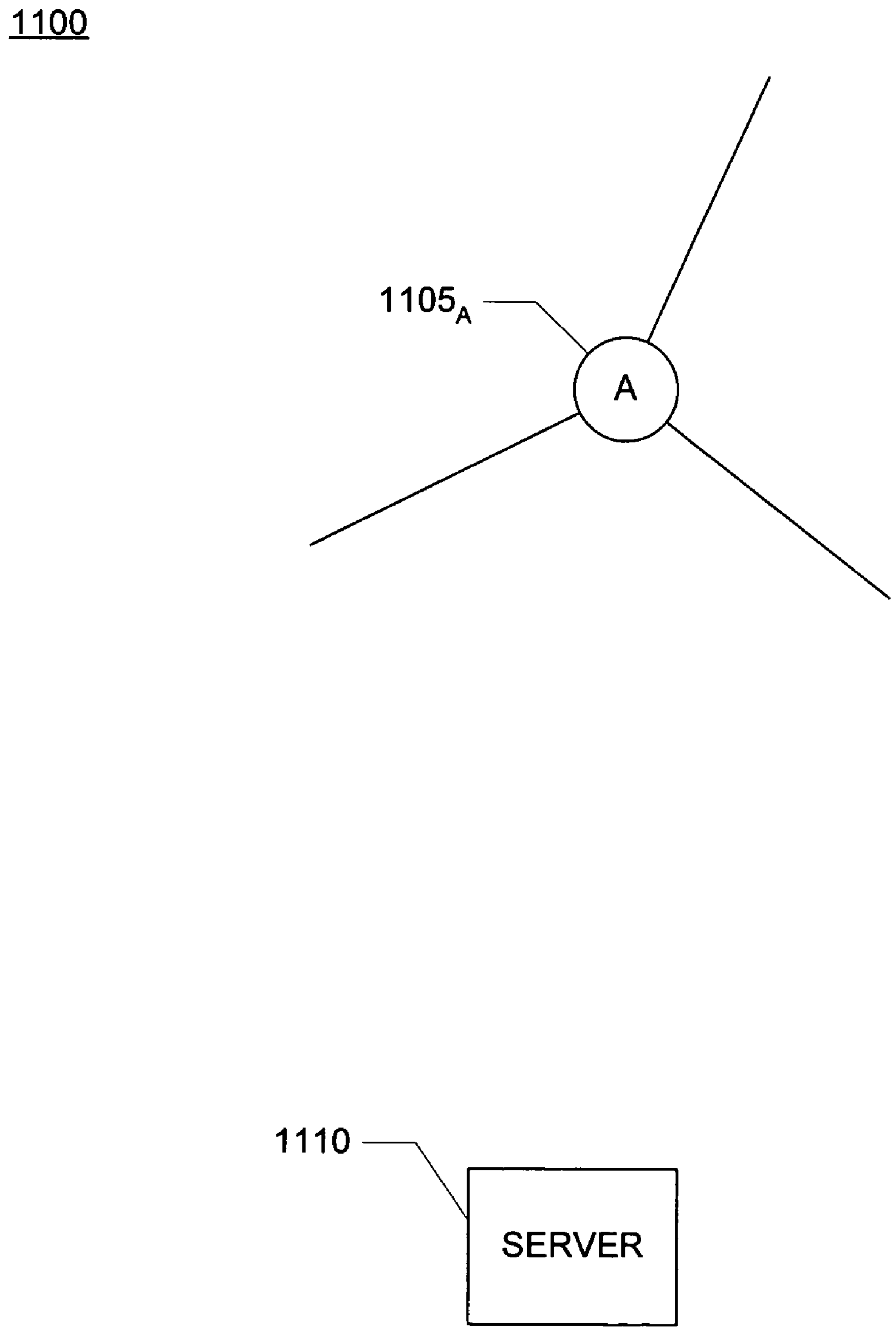


FIG. 12

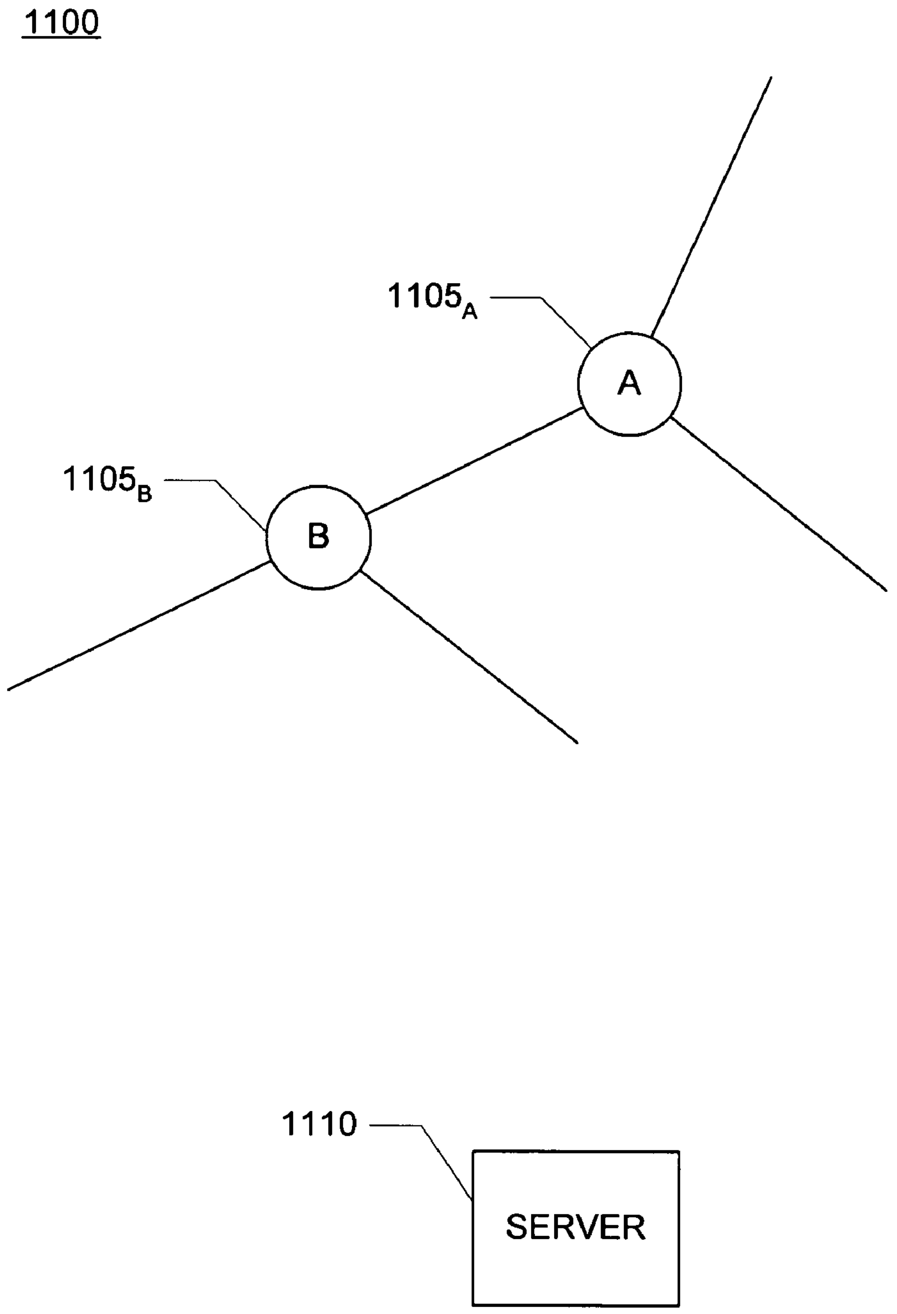


FIG. 13

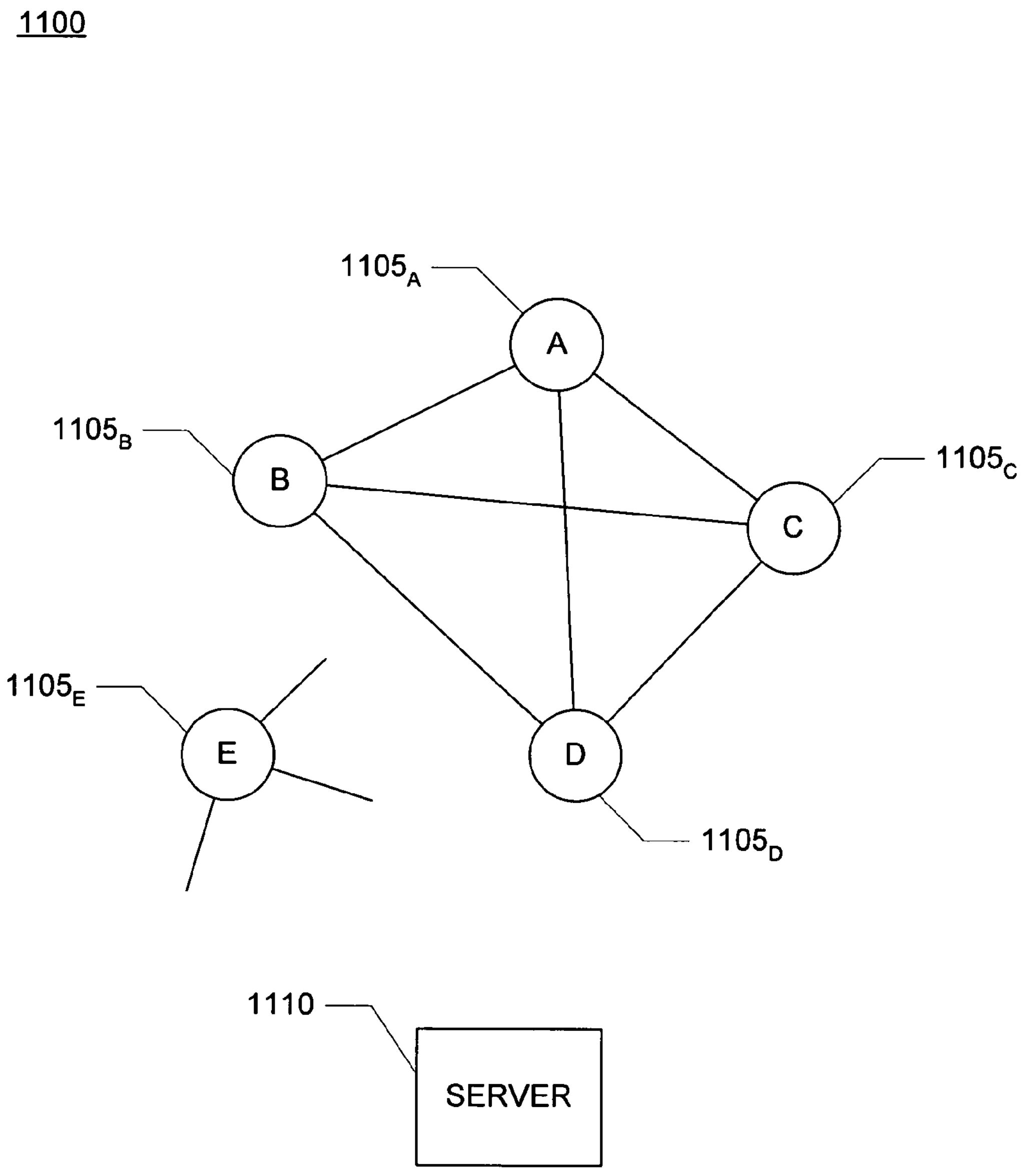


FIG. 14

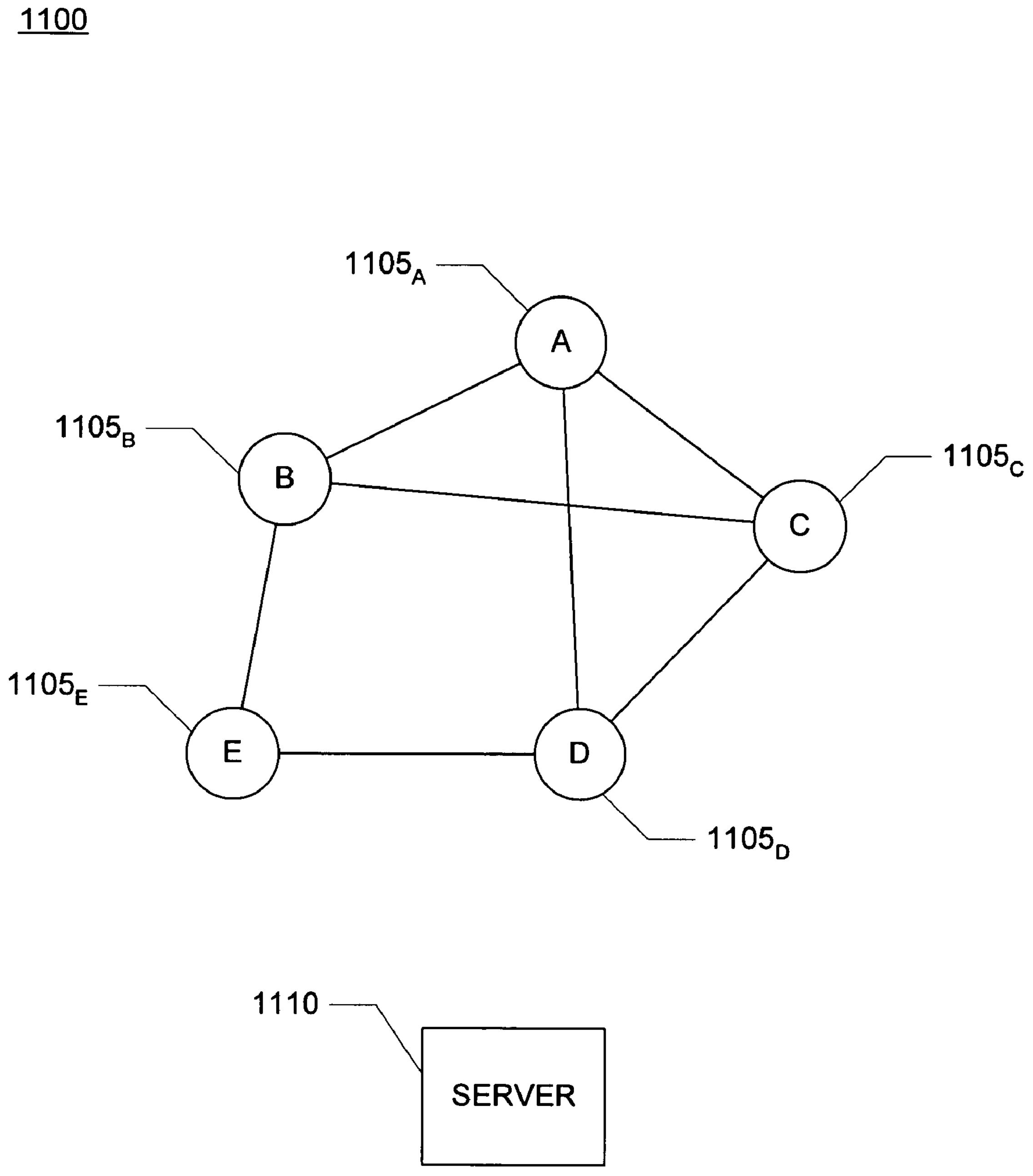


FIG. 15

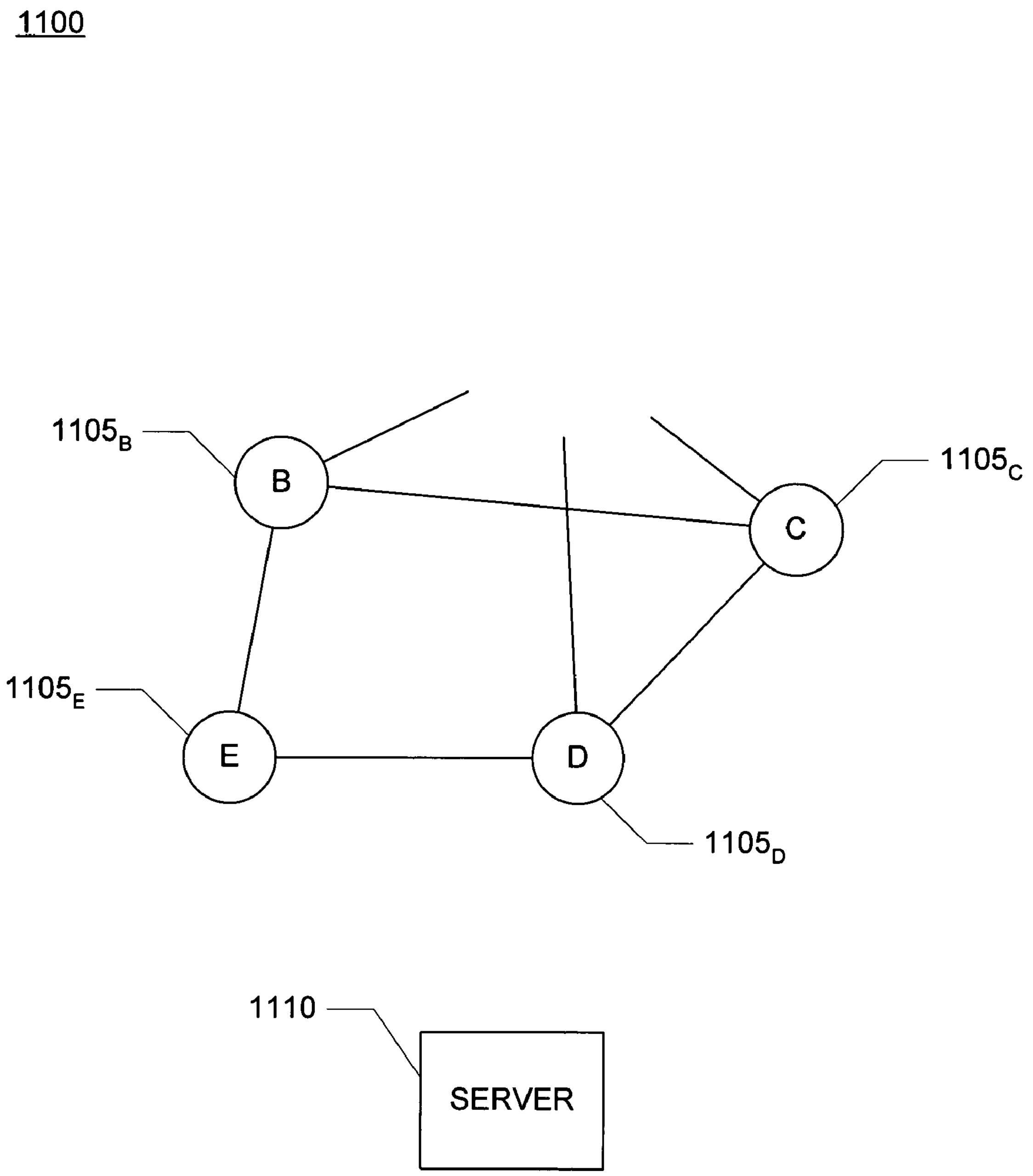


FIG. 16

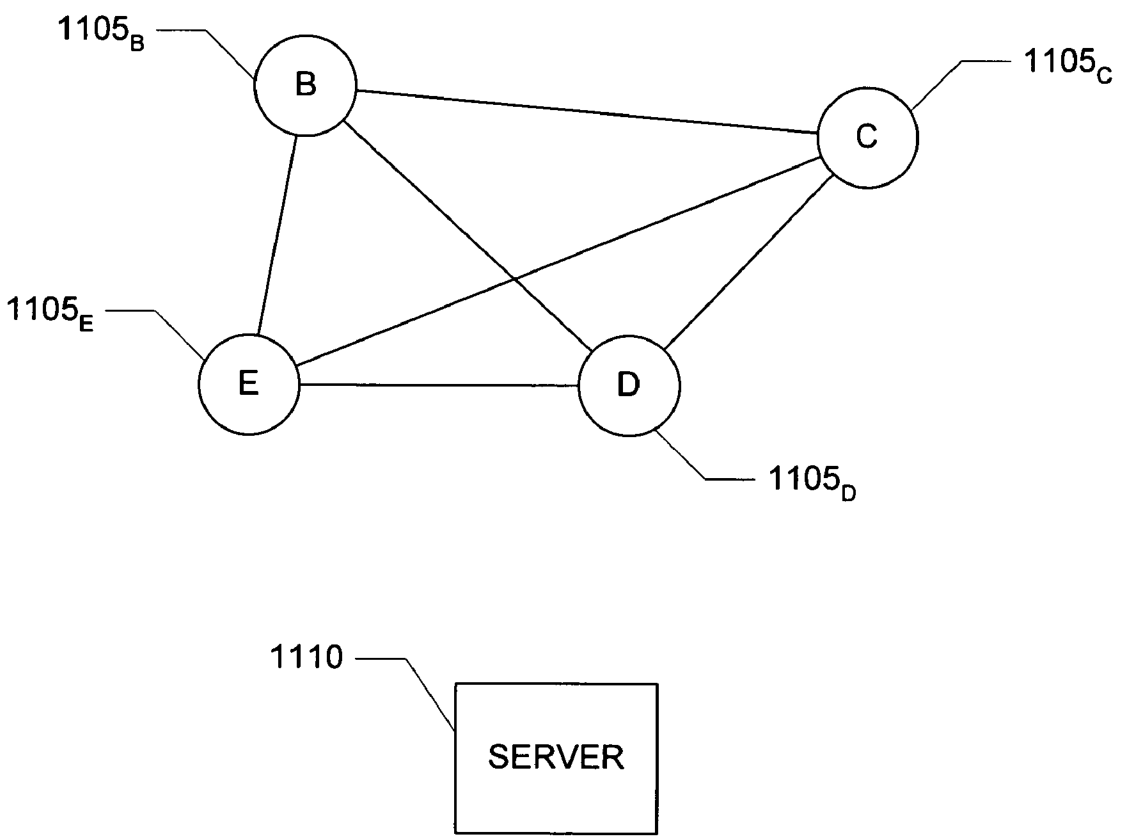


FIG. 17

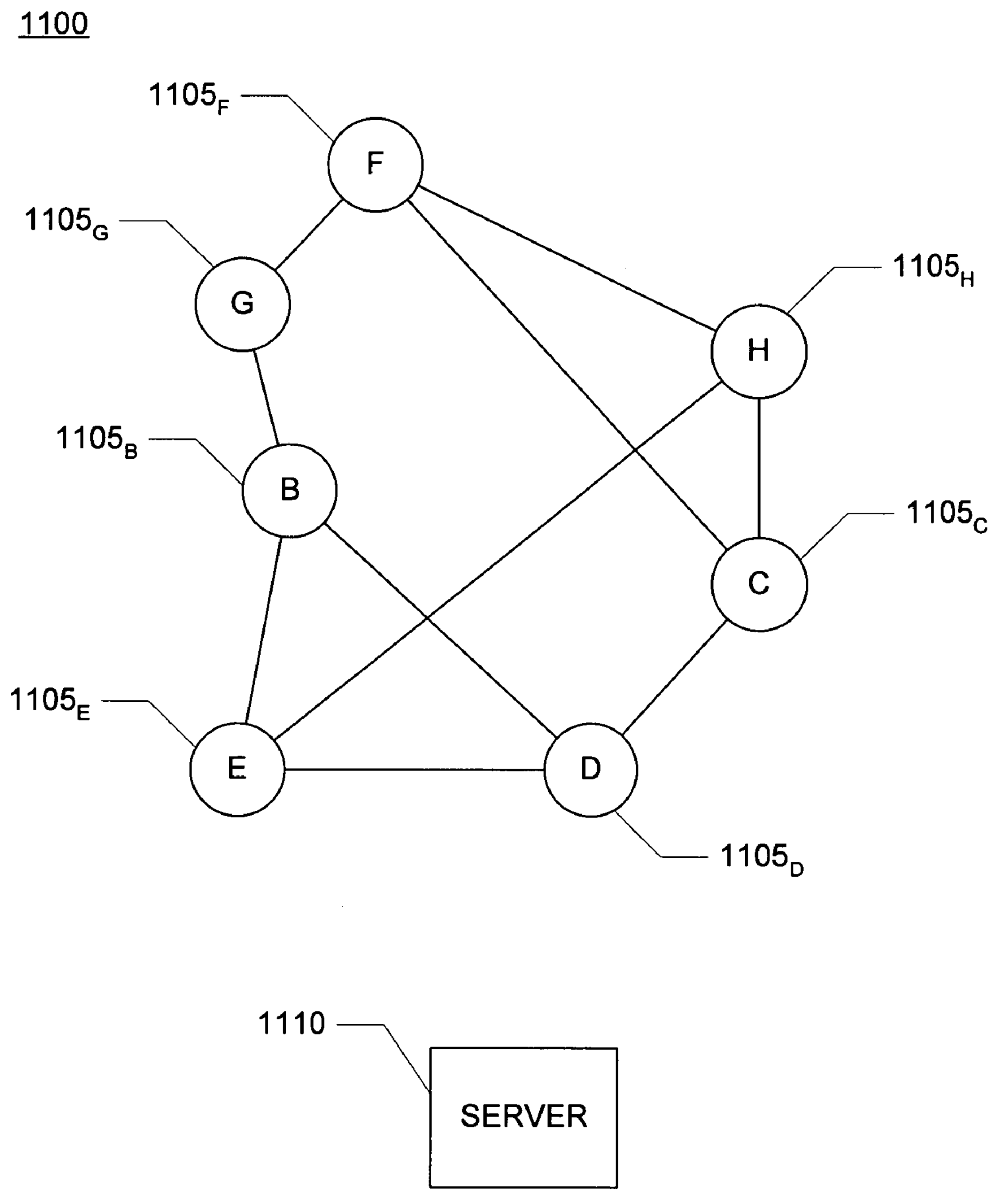
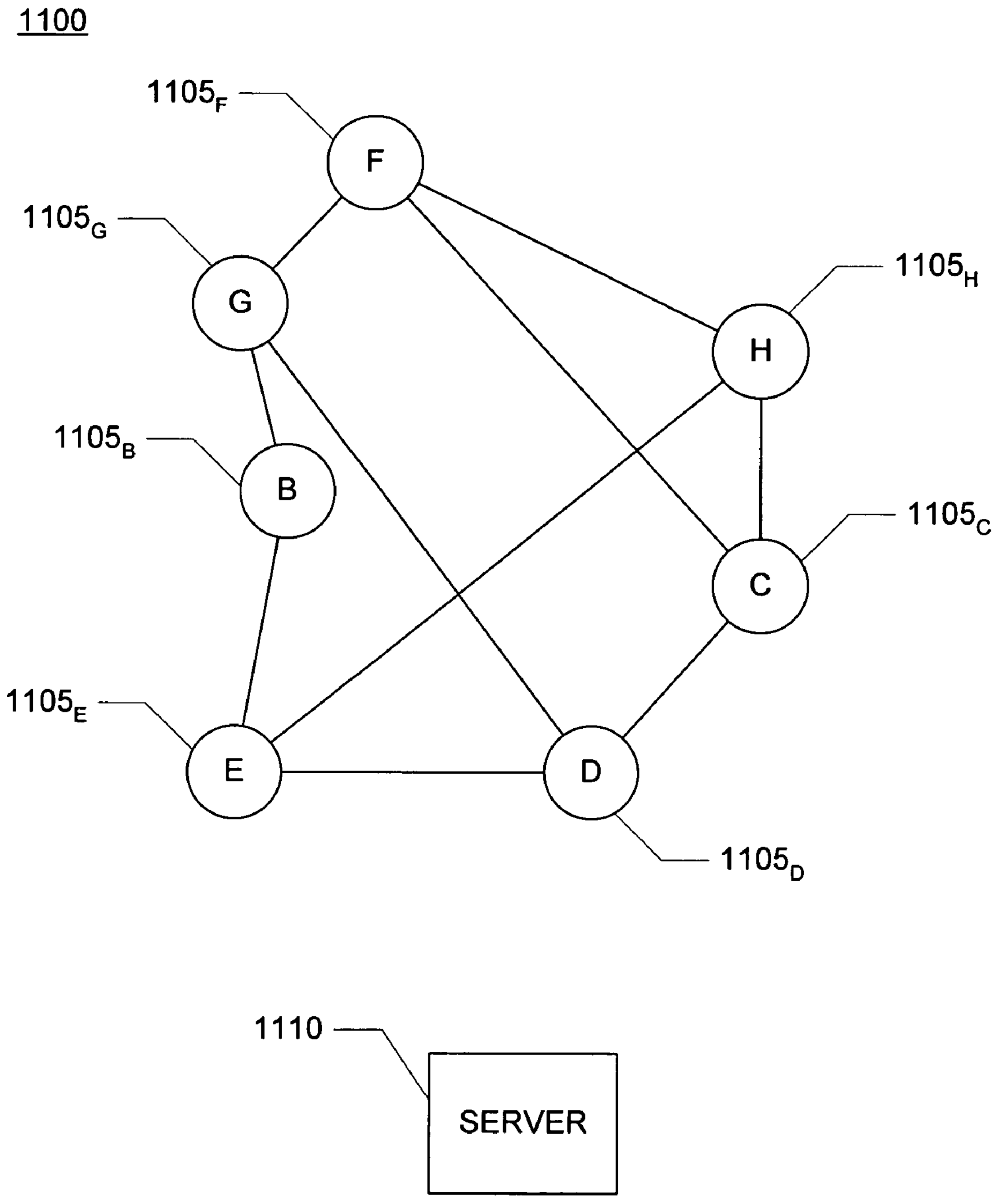

1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 18



1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 19

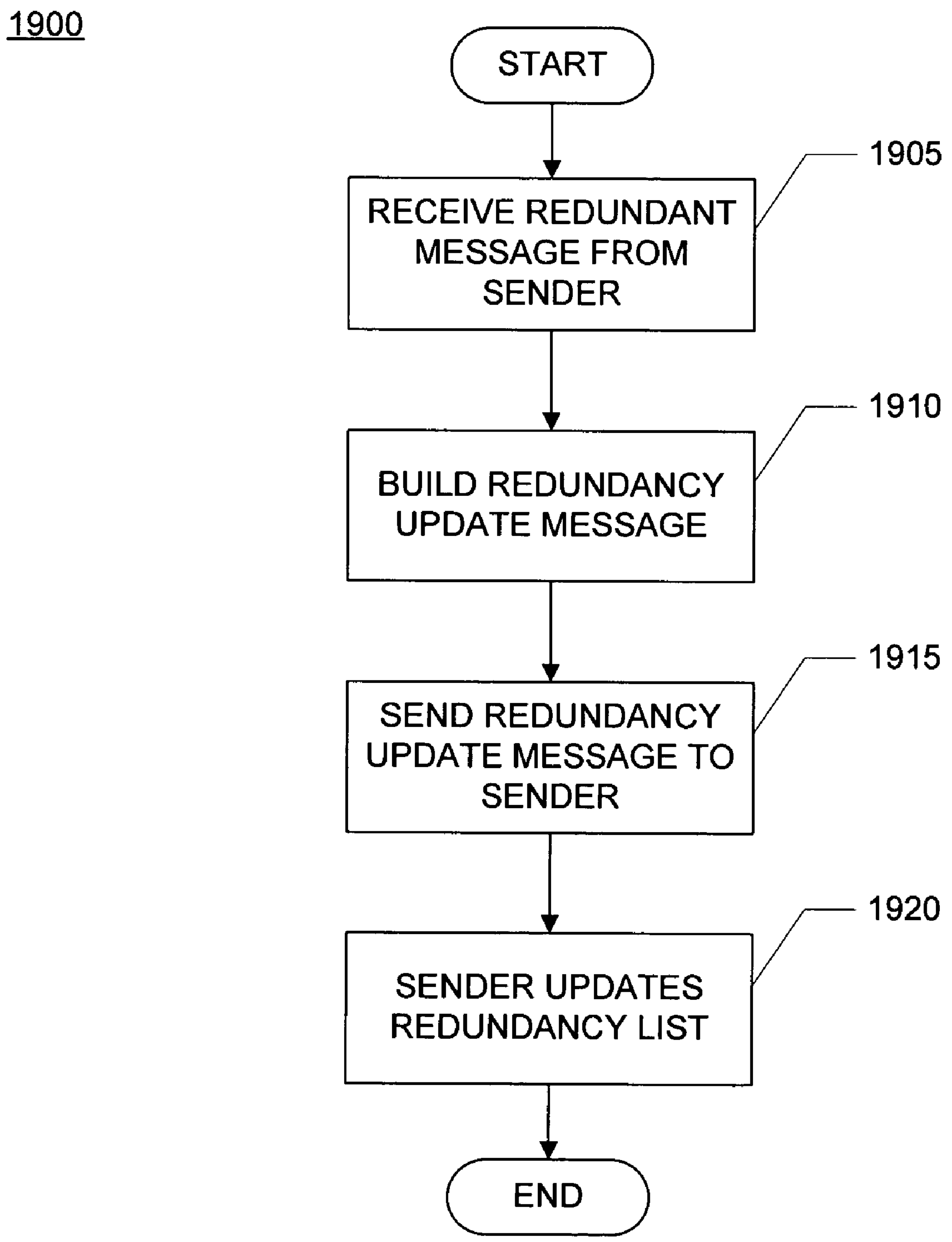

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

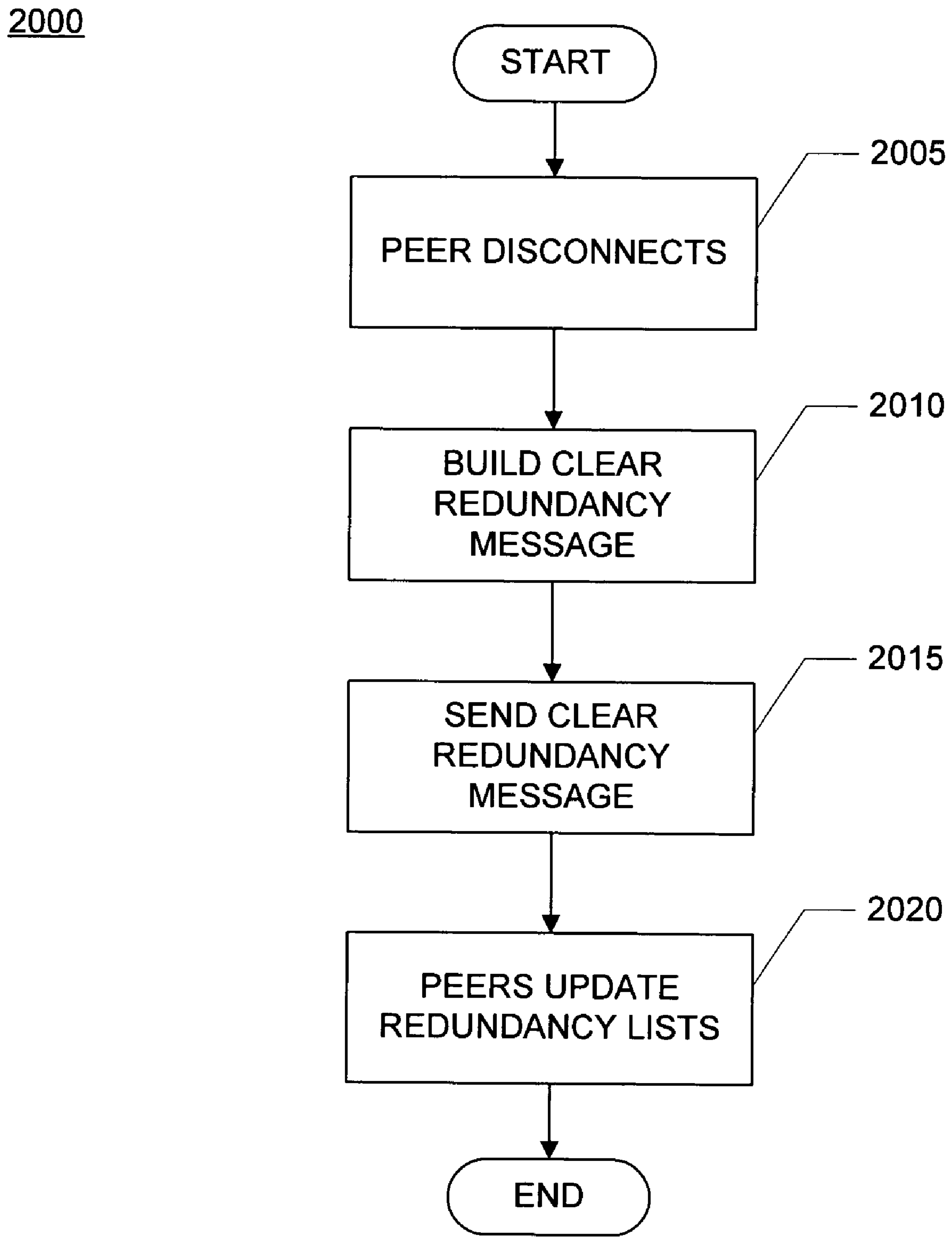

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

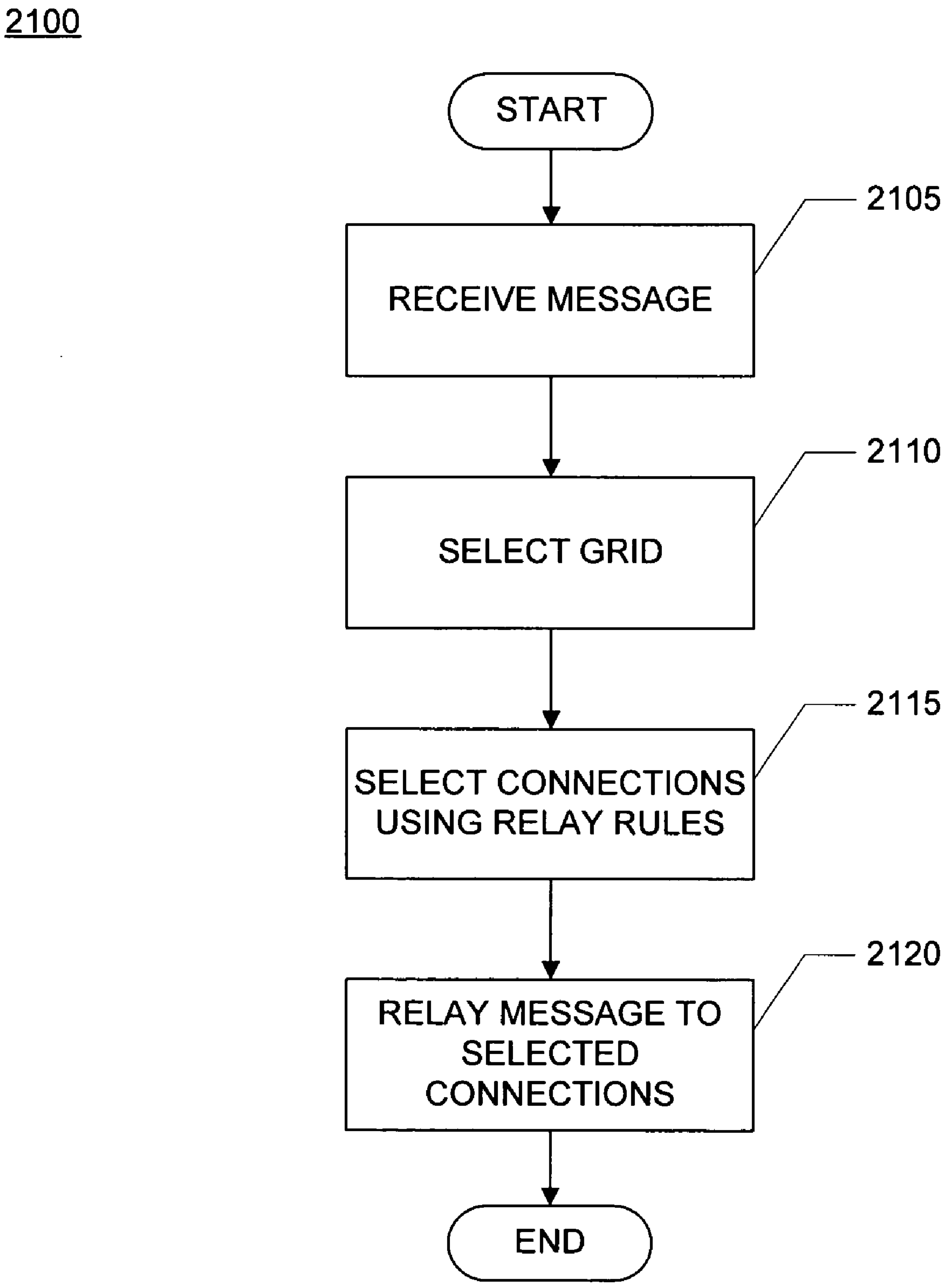

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

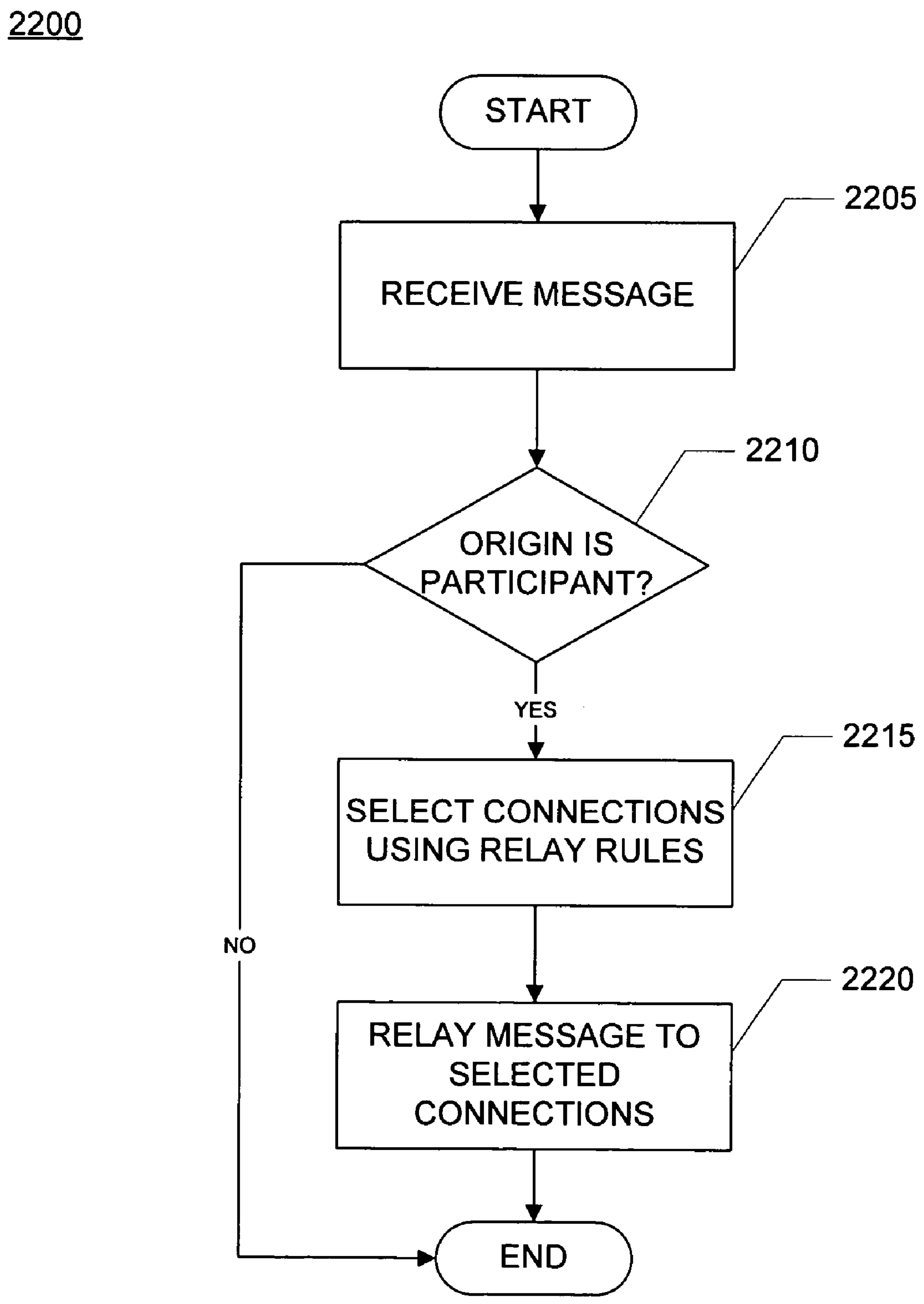

2200
START
RECEIVE MESSAGE
2205
ORIGIN IS PARTICIPANT?
2210
YES
NO
SELECT CONNECTIONS USING RELAY RULES
2215
RELAY MESSAGE TO SELECTED CONNECTIONS
2220
END

FIG. 23

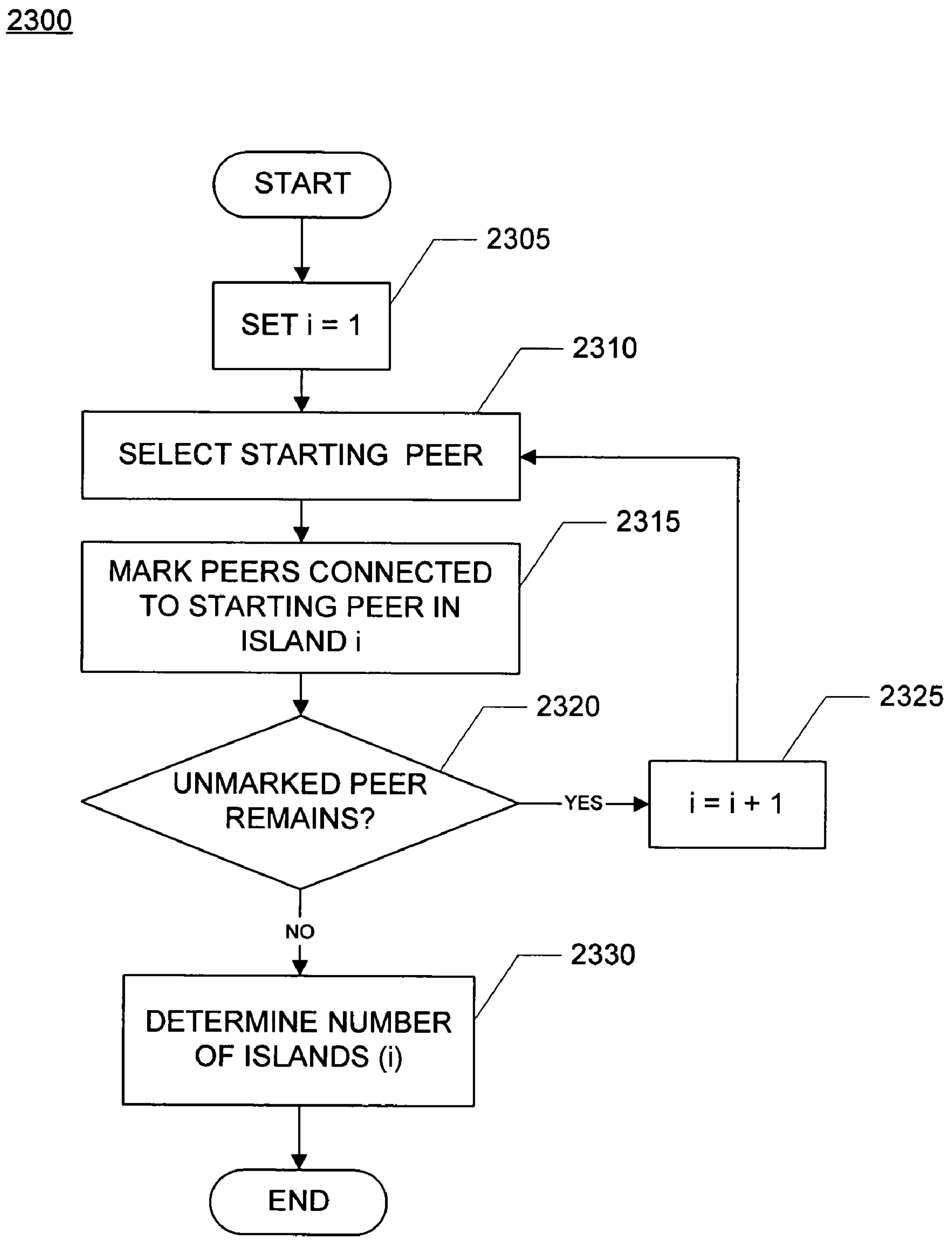

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

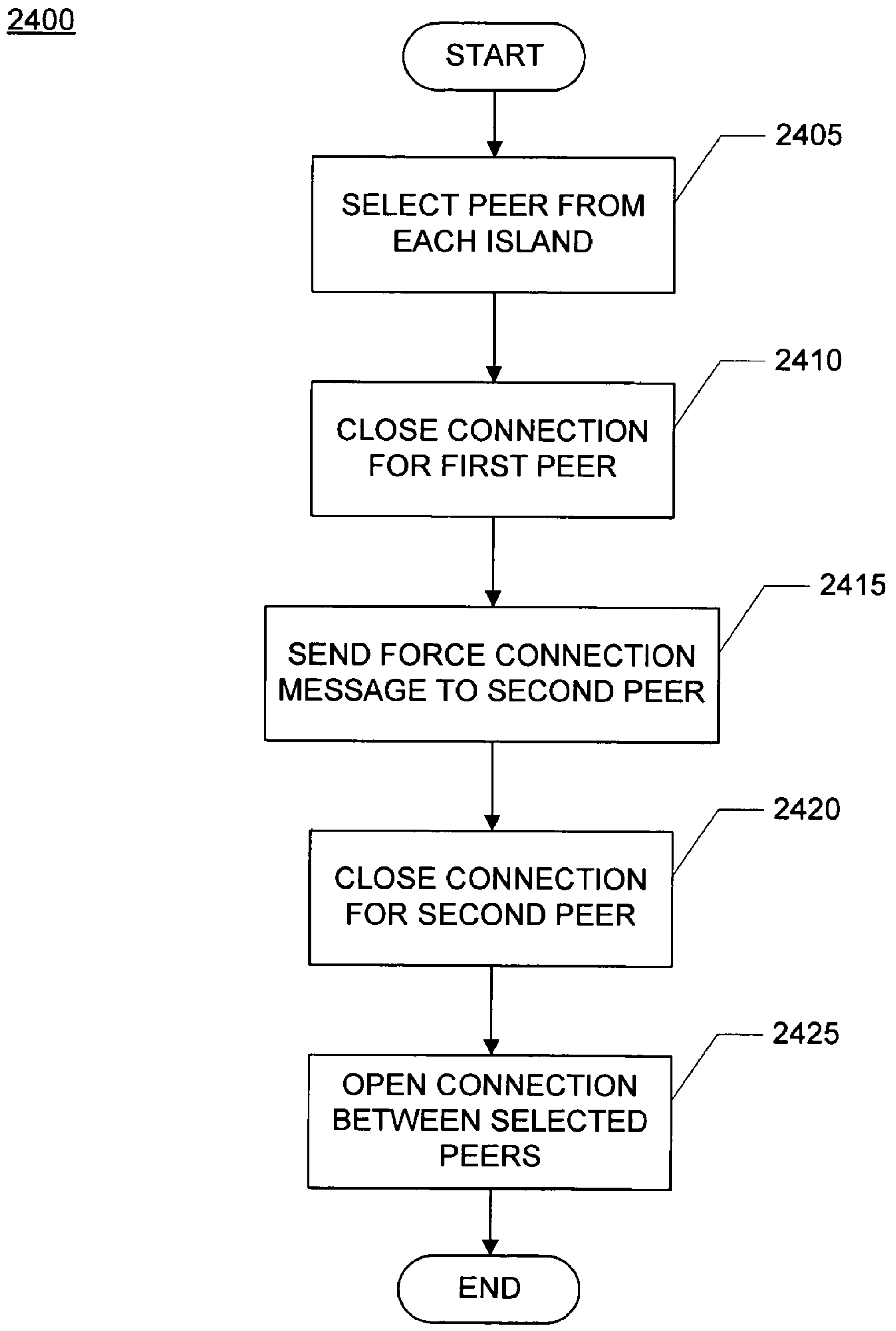

2400
START
SELECT PEER FROM EACH ISLAND
2405
CLOSE CONNECTION FOR FIRST PEER
2410
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2415
CLOSE CONNECTION FOR SECOND PEER
2420
OPEN CONNECTION BETWEEN SELECTED PEERS
2425
END

FIG. 25

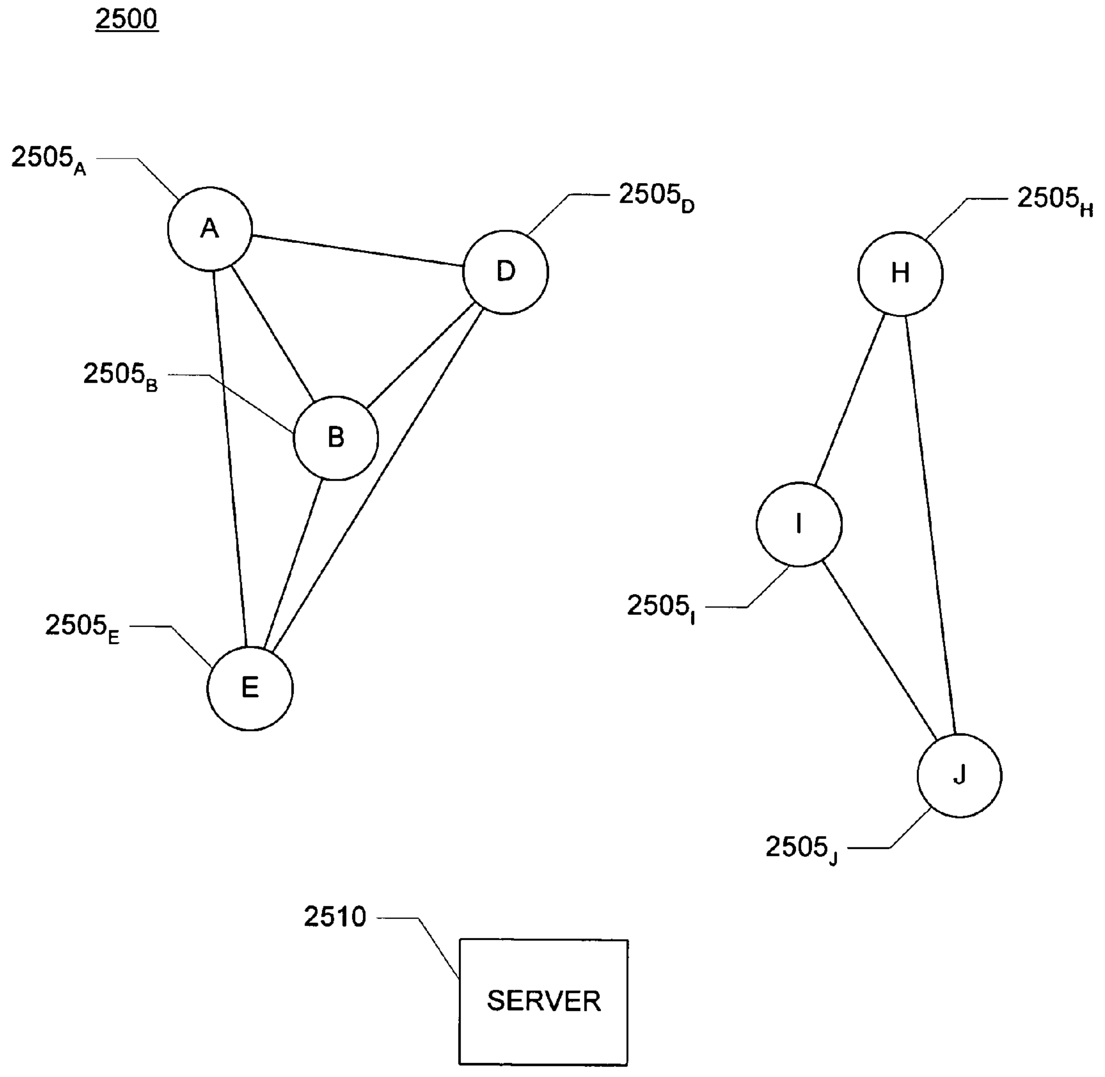

2500
2505_A
A
2505_D
D
2505_H
H
2505_B
B
I
2505_I
2505_E
E
J
2505_J
2510
SERVER

FIG. 26

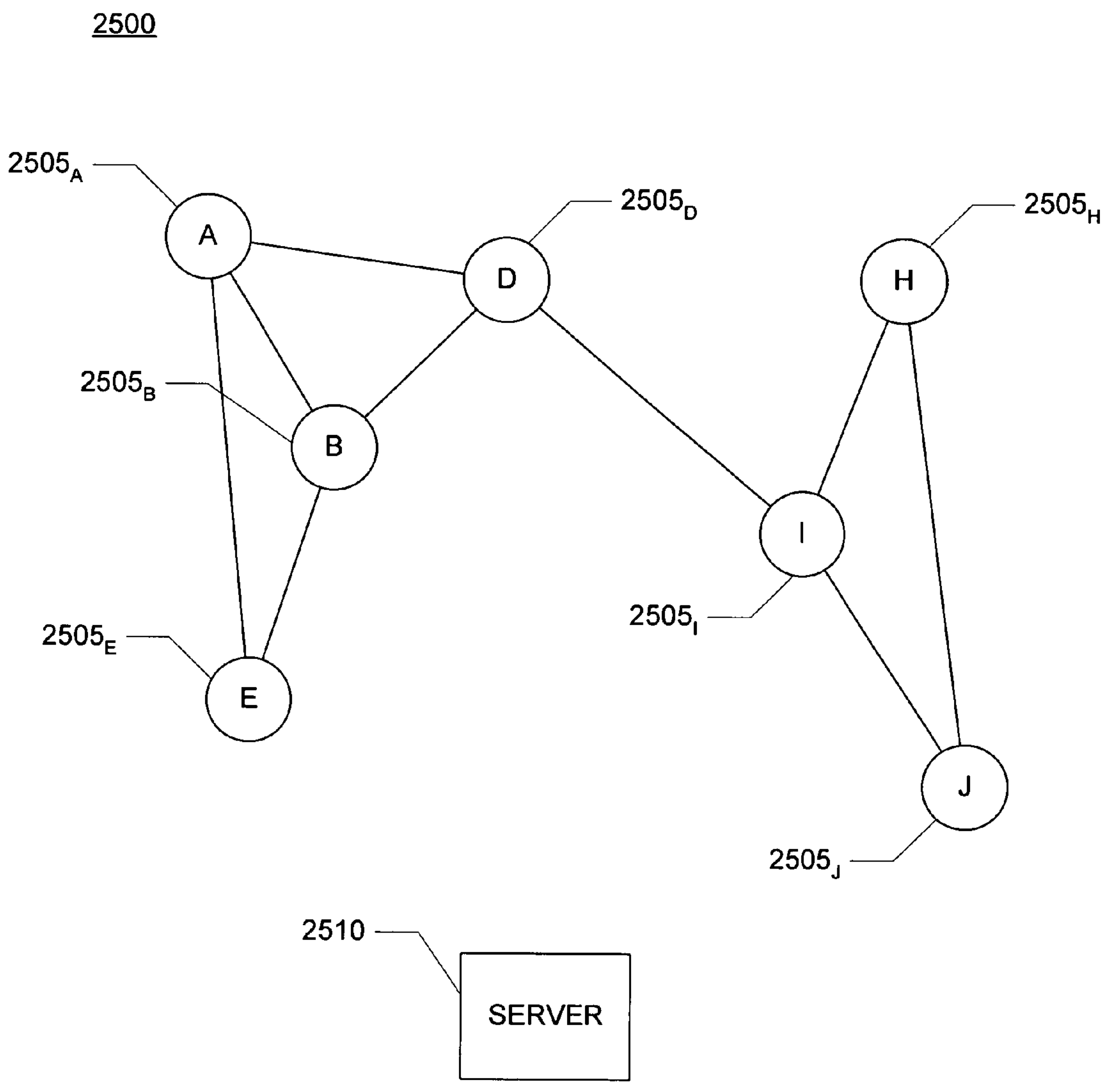

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 27

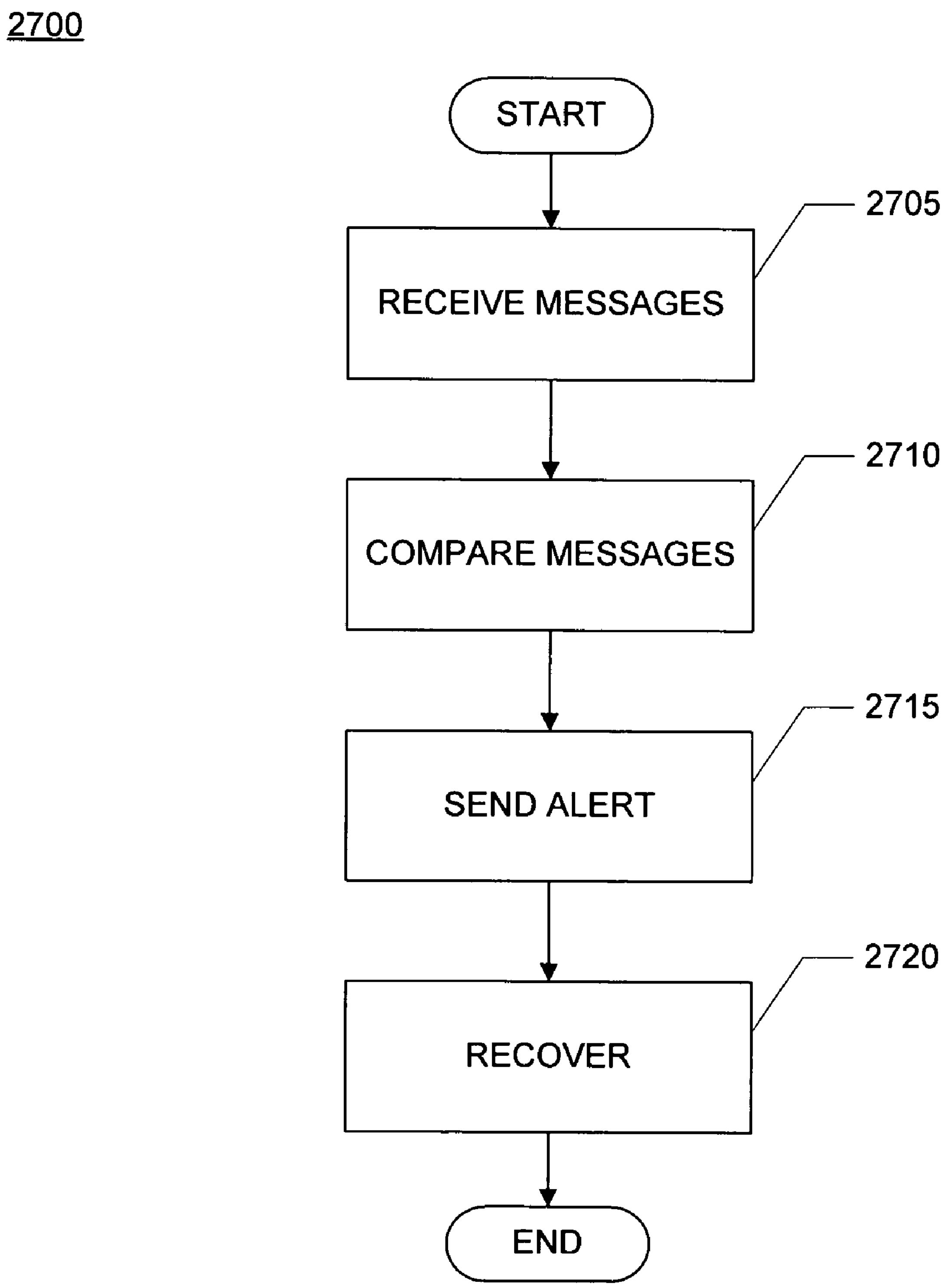

2700
START
2705
RECEIVE MESSAGES
2710
COMPARE MESSAGES
2715
SEND ALERT
2720
RECOVER
END

FIG. 28

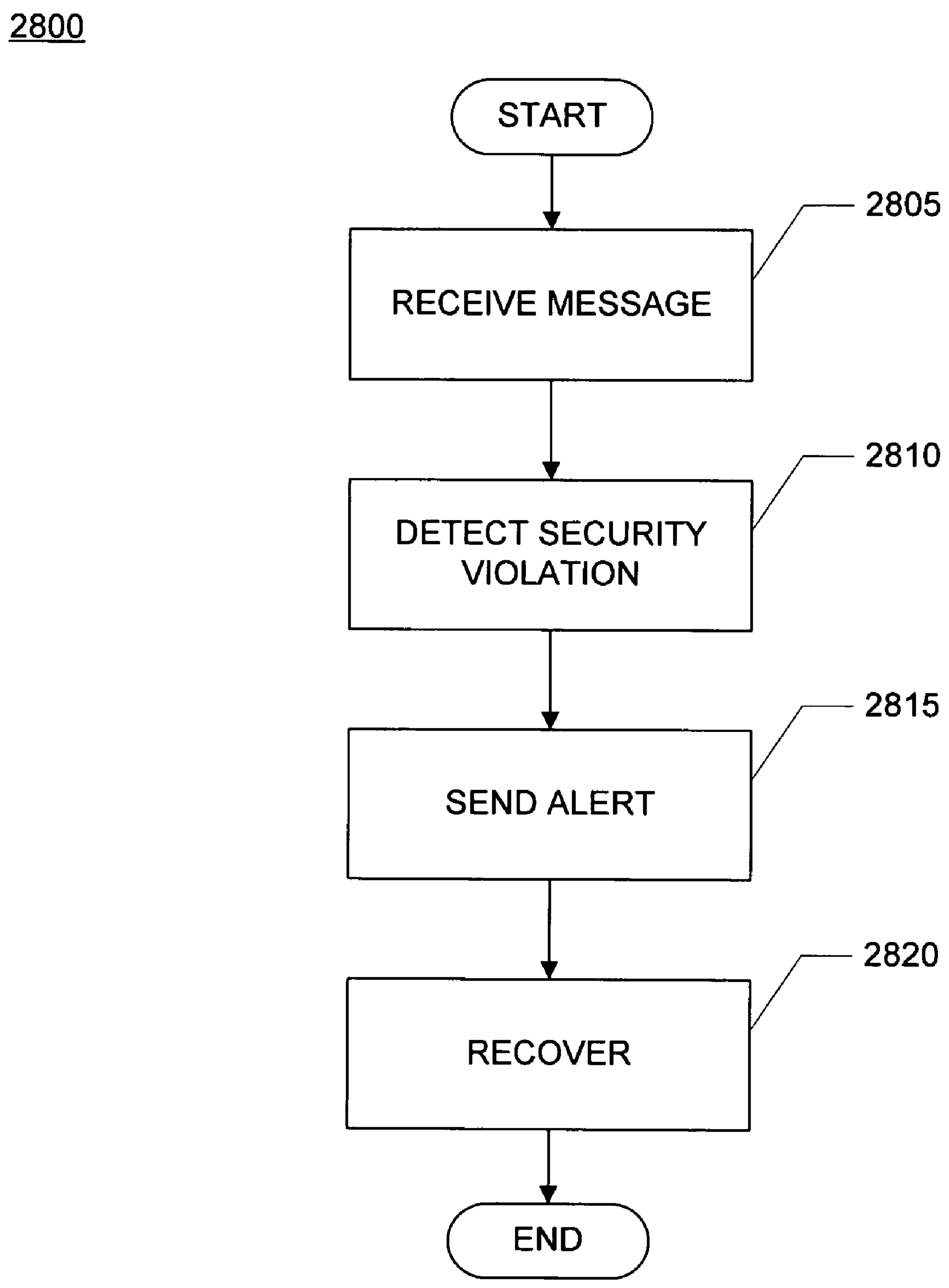

2800
START
RECEIVE MESSAGE
2805
DETECT SECURITY VIOLATION
2810
SEND ALERT
2815
RECOVER
2820
END

FIG. 29

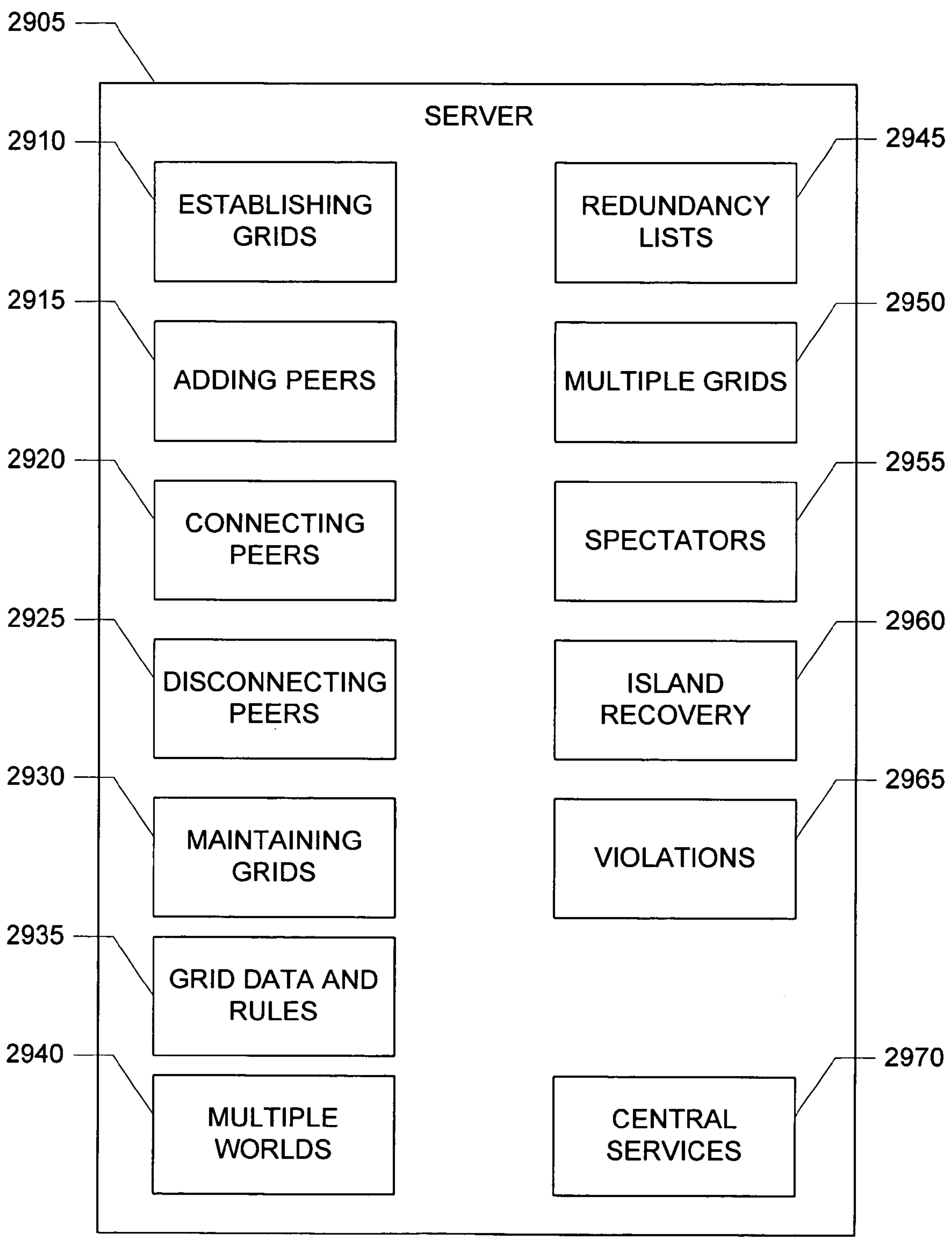

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

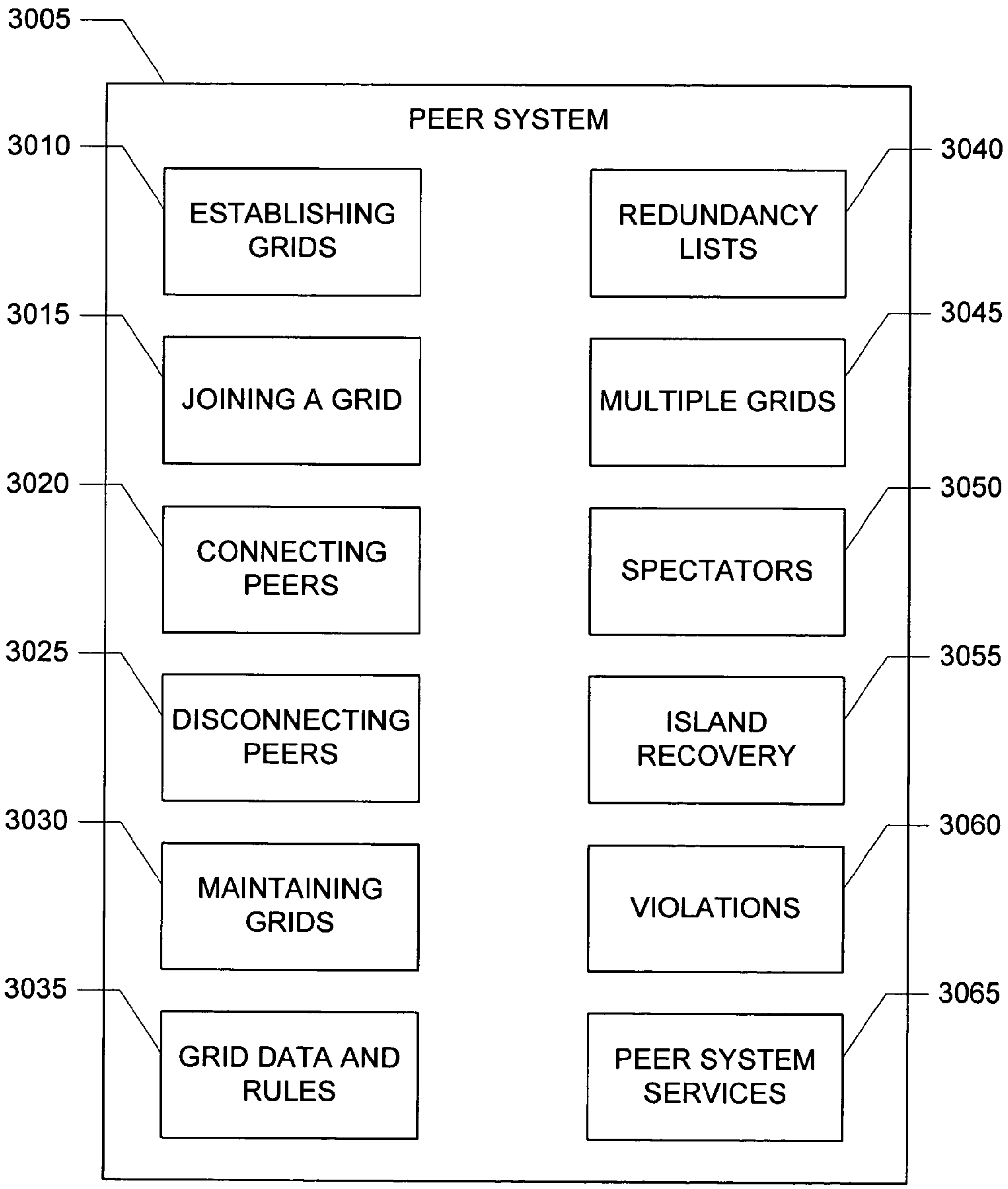

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

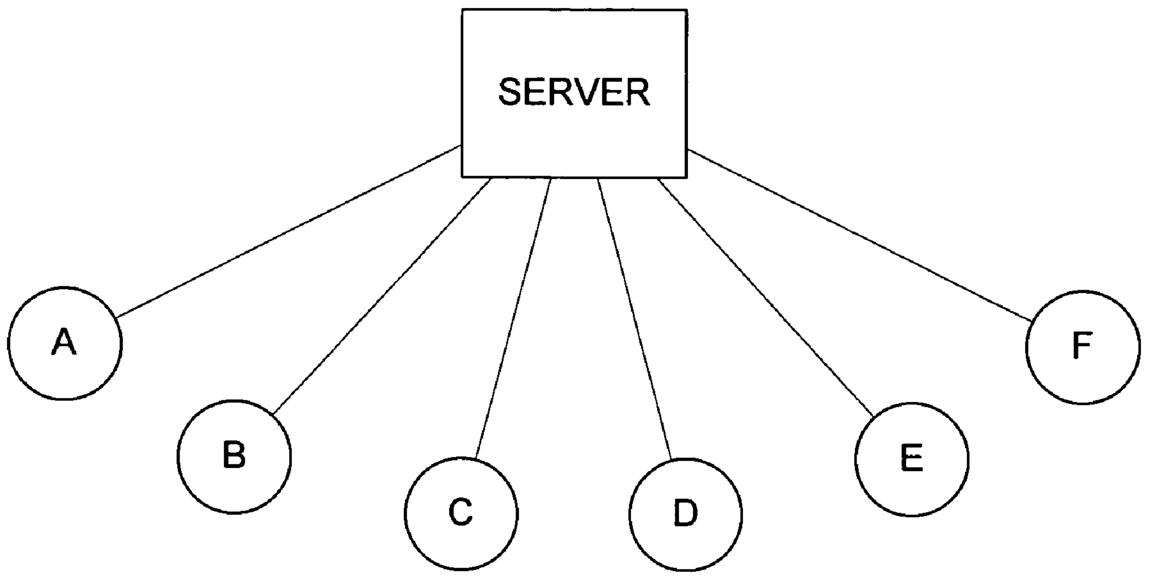
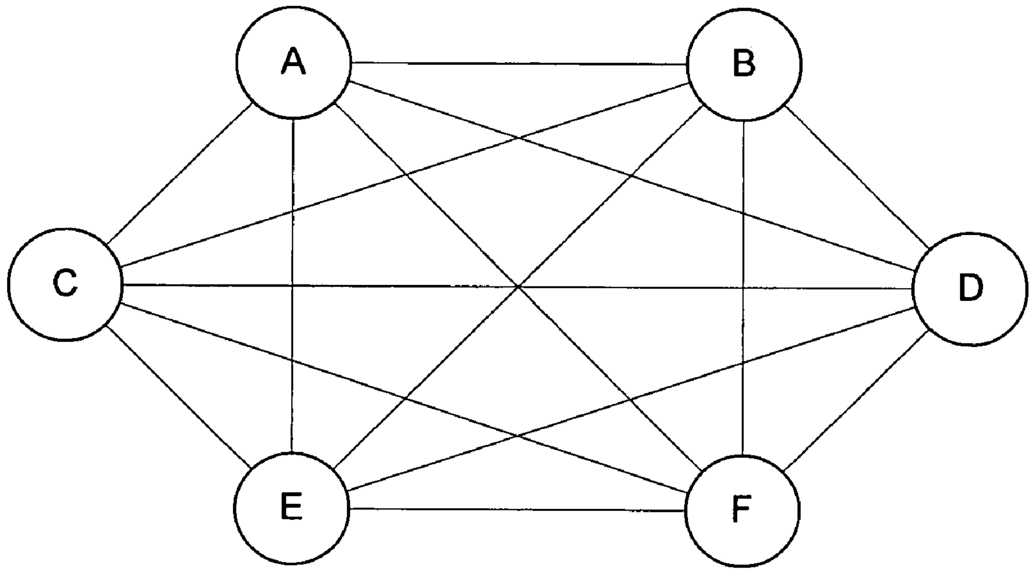


FIG. 31B



SPECTATORS IN A PEER-TO-PEER RELAY NETWORK

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

This application is related to the U.S. application Ser. No. 10/700,798, filed on Nov. 3, 2003, Ser. No. 10/701,302, filed on Nov. 3, 2003, Ser. No. 10/701,014, filed on Nov. 3,2003, Ser. No. 10/701,298, filed on Nov. 3, 2003, and Ser. No. 10/700,797, filed on Nov. 3, 2003.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N−1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system in said peer-to-peer relay network is connected to a number of other peer systems in said peer-to-peer relay network that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, each peer system in said peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules, each peer system is a participant or a spectator, at least one peer system is a participant, at least one peer system is a spectator, a participant is configured to generate data to be relayed in said peer-to-peer relay network, and a spectator is configured to relay data generated by a participant.

In one implementation, a method of relaying data in a peer-to-peer relay network includes: receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network, wherein said data has associated information identifying the origin peer system that generated said data; confirming said origin peer system is permitted to send data to be relayed through said peer-to-peer relay network; applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and relaying said data to any peer systems selected by applying said set of one or more relay rules; wherein each peer system in said peer-to-peer relay network is a participant or a spectator.

In one implementation, a peer system in a peer-to-peer relay network includes: means for receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network, wherein said data has associated information identifying the origin peer system that generated said data; means for confirming said origin peer system is permitted to send data to be relayed through said peer-to-peer relay network; means for applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and means for relaying said data to any peer systems selected by applying said set of one or more relay rules; wherein each peer system in said peer-to-peer relay network is a participant or a spectator.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent application Ser. Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed Jul. 31, 2002, and 10/359,359 ("Multi-User Application Programming Interface"), filed Feb. 4, 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system **105**$_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11**-**18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins, peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11**-**18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can

become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection

message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer.

For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A peer-to-peer relay network, comprising:

a plurality of N peer systems, wherein each peer system is either a participant or a spectator and the peer-to-peer network includes at least one participant and at least one spectator;

wherein each peer system in said peer-to-peer relay network is connected to a number of other peer systems in said peer-to-peer relay network that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, each peer system in said peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules,

wherein a participant is configured to generate data to be relayed in said peer-to-peer relay network, and a spectator is configured to relay data generated by a participant, and

wherein the spectator is not authorized to generate new data to be relayed in the peer-to-peer network and the spectator is not authorized to send the new data to be relayed throughout the peer-to-peer network.

2. The peer-to-peer relay network of claim **1**, further comprising:

a server connected to each peer system.

3. The peer-to-peer relay network of claim **1**, wherein:

said at least one participant is playing an online game.

4. The peer-to-peer relay network of claim **1**, wherein:

said at least one participant is performing.

5. The peer-to-peer relay network of claim **4**, wherein:

said performing is playing music.

6. The peer-to-peer relay network of claim **1**, wherein:

said at least one participant is teaching.

7. The peer-to-peer relay network of claim **1**, wherein:

at least two peer systems are participants, and each participant has a connection to at least one other participant.

8. The peer-to-peer relay network of claim **1**, wherein:

each peer system is configured not to relay data generated by a spectator.

9. The peer-to-peer relay network of claim **1**, wherein:

at least one spectator is a conditional spectator, a conditional spectator is configured to request permission to send data generated by the conditional spectator to other peer systems to be relayed throughout said peer-to-peer relay network, each peer system is configured to relay data generated by a conditional spectator if that conditional spectator has received permission to send that data.

10. The peer-to-peer relay network of claim **1**, wherein:

at least one peer system is a network-enabled game console.

11. The peer-to-peer relay network of claim **1**, wherein:

at least two peer systems are connected through the Internet.

12. A method of relaying data in a peer-to-peer relay network, comprising:

receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network, wherein said data has associated information identifying the origin peer system that generated said data;

confirming said origin peer system is permitted to send data to be relayed through said peer-to-peer relay network;

applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and

relaying said data to any peer systems selected by applying said set of one or more relay rules;

wherein each peer system in said peer-to-peer relay network is a participant or a spectator, and wherein the spectator is not authorized to generate new data to be relayed in the peer-to-peer network and the spectator is not authorized to send the new data to be relayed throughout the peer-to-peer network.

13. The method of claim **12**, wherein:

each peer system stores a connection limit defining a number of other peer systems up to which a peer system is permitted to connect in that peer-to-peer relay network, and each peer system stores a set of one or more relay rules defining how a peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

14. A peer system in a peer-to-peer relay network, comprising:

means for receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network, wherein said data has associated information identifying the origin peer system that generated said data;

means for confirming said origin peer system is permitted to send data to be relayed through said peer-to-peer relay network;

means for applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and

means for relaying said data to any peer systems selected by applying said set of one or more relay rules;

wherein each peer system in said peer-to-peer relay network is a participant or a spectator, and

wherein the spectator is not authorized to generate new data to be relayed in the peer-to-peer network and the spectator is not authorized to send the new data to be relayed throughout the peer-to-peer network.

15. The peer system of claim **14**, wherein:

said peer system stores a connection limit defining a number of other peer systems up to which said peer system is permitted to connect in that peer-to-peer relay network, and said peer system stores a set of one or more relay rules defining how said peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

16. A computer program product, comprising a computer usable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a peer system in a peer-to-peer relay network, said method comprising step to:

process received data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network, wherein said data has associated information identifying the origin peer system that generated said data;

confirm said origin peer system is permitted to send data to be relayed through said peer-to-peer relay network;

apply a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and

relay said data to any peer systems selected by applying said set of one or more relay rules;

wherein each peer system in said peer-to-peer relay network is a participant or a spectator, and

wherein the spectator is not authorized to generate new data to be relayed in the peer-to-peer network and the spectator is not authorized to send the new data to be relayed throughout the peer-to-peer network.

17. The computer program of claim **16**, wherein:

said peer system stores a connection limit defining a number of other peer systems up to which said peer system is permitted to connect in that peer-to-peer relay network, and said peer system stores a set of one or more relay rules defining how said peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 7,610,402 B2	Page 1 of 1
APPLICATION NO.	: 10/700777	
DATED	: October 27, 2009	
INVENTOR(S)	: Glen Van Datta	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1756 days.

Signed and Sealed this

Twelfth Day of October, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

EXHIBIT 85

US007610505B2

(12) **United States Patent**
Datta et al.

(10) **Patent No.: US 7,610,505 B2**
(45) **Date of Patent: Oct. 27, 2009**

(54) VIOLATIONS IN A PEER-TO-PEER RELAY NETWORK

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Foster City, 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.: **12/011,108**

(22) Filed: **Jan. 24, 2008**

(65) **Prior Publication Data**

US 2008/0147854 A1 Jun. 19, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/700,797, filed on Nov. 3, 2003, now Pat. No. 7,392,422.

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/4**; 714/49; 709/224; 713/181; 726/22

(58) **Field of Classification Search** 714/4, 714/49, 716, 819, 820; 713/179, 181; 726/10, 726/22, 23, 24; 702/185
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,325	A	8/1996	Denny et al.	709/236
5,768,382	A	6/1998	Schneier et al.	380/251
5,835,726	A *	11/1998	Shwed et al.	709/229
6,212,633	B1 *	4/2001	Levy et al.	713/153
6,327,630	B1	12/2001	Carroll et al.	719/314

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999

(Continued)

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

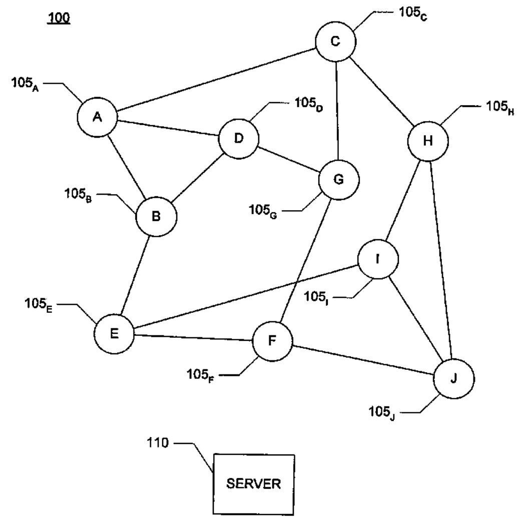
(Continued)

Primary Examiner—Robert Beausoliel
Assistant Examiner—Elmira Mehrmanesh
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of detecting and recovering from violations in a peer-to-peer relay network includes: receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network detecting a violation in said received message; and sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

22 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

6,405,104	B1	6/2002	Dougherty	700/292
6,640,241	B1 *	10/2003	Ozzie et al.	709/204
6,779,017	B1 *	8/2004	Lamberton et al.	709/203
6,829,634	B1	12/2004	Holt et al.	709/204
6,978,294	B1 *	12/2005	Adams et al.	709/217
7,043,641	B1 *	5/2006	Martinek et al.	713/187
7,127,613	B2 *	10/2006	Pabla et al.	713/171
7,168,089	B2 *	1/2007	Nguyen et al.	726/4
7,203,841	B2 *	4/2007	Jackson et al.	713/187
7,321,928	B2 *	1/2008	Feltin et al.	709/223
7,392,375	B2 *	6/2008	Bartram et al.	713/152
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0107786	A1 *	8/2002	Lehmann-Haupt et al.	705/37
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1	12/2002	Traversat et al.	
2003/0028585	A1 *	2/2003	Yeager et al.	709/201
2003/0055892	A1	3/2003	Huitema et al.	
2003/0126245	A1 *	7/2003	Feltin et al.	709/223
2003/0229779	A1	12/2003	Morais et al.	713/153
2003/0229789	A1	12/2003	Morais et al.	713/171
2004/0088369	A1 *	5/2004	Yeager et al.	709/217
2004/0243665	A1 *	12/2004	Markki et al.	709/201
2005/0020354	A1	1/2005	Nguyen et al.	463/25
2005/0086287	A1	4/2005	Datta	709/201
2005/0086288	A1	4/2005	Datta et al.	709/201
2005/0086329	A1	4/2005	Datta et al.	709/220
2005/0086350	A1	4/2005	Mai	709/230
2005/0086369	A1	4/2005	Mai et al.	709/239

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" Globecom 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups", Proceedings of the 7_{TH} ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. Conf. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

Baughman et al., Cheat-proof playout for centralized and distributed online games, INFOCOM2001. Twentieth Annual Joint Conference of the IEEE Computer and Communciations Societies. Proceedings. IEEE Publication Date: Apr. 22-26, 2001, On pp. 104-113, vol. 1.

* cited by examiner

FIG. 1

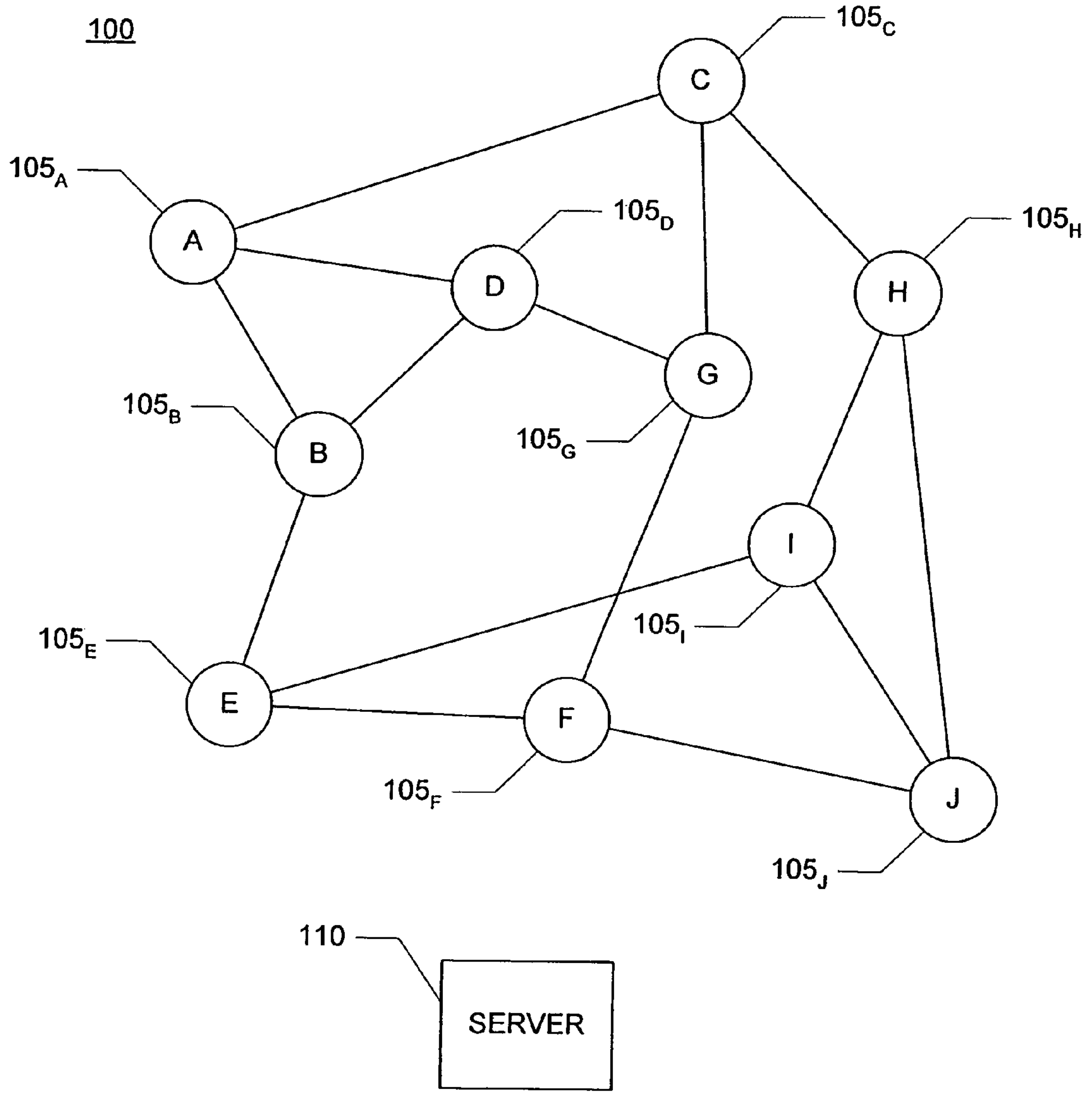

100
105_C
C
105_A
A
105_D
D
105_H
H
G
105_B
B
105_G
I
105_I
105_E
E
F
105_F
J
105_J
110
SERVER

FIG. 2

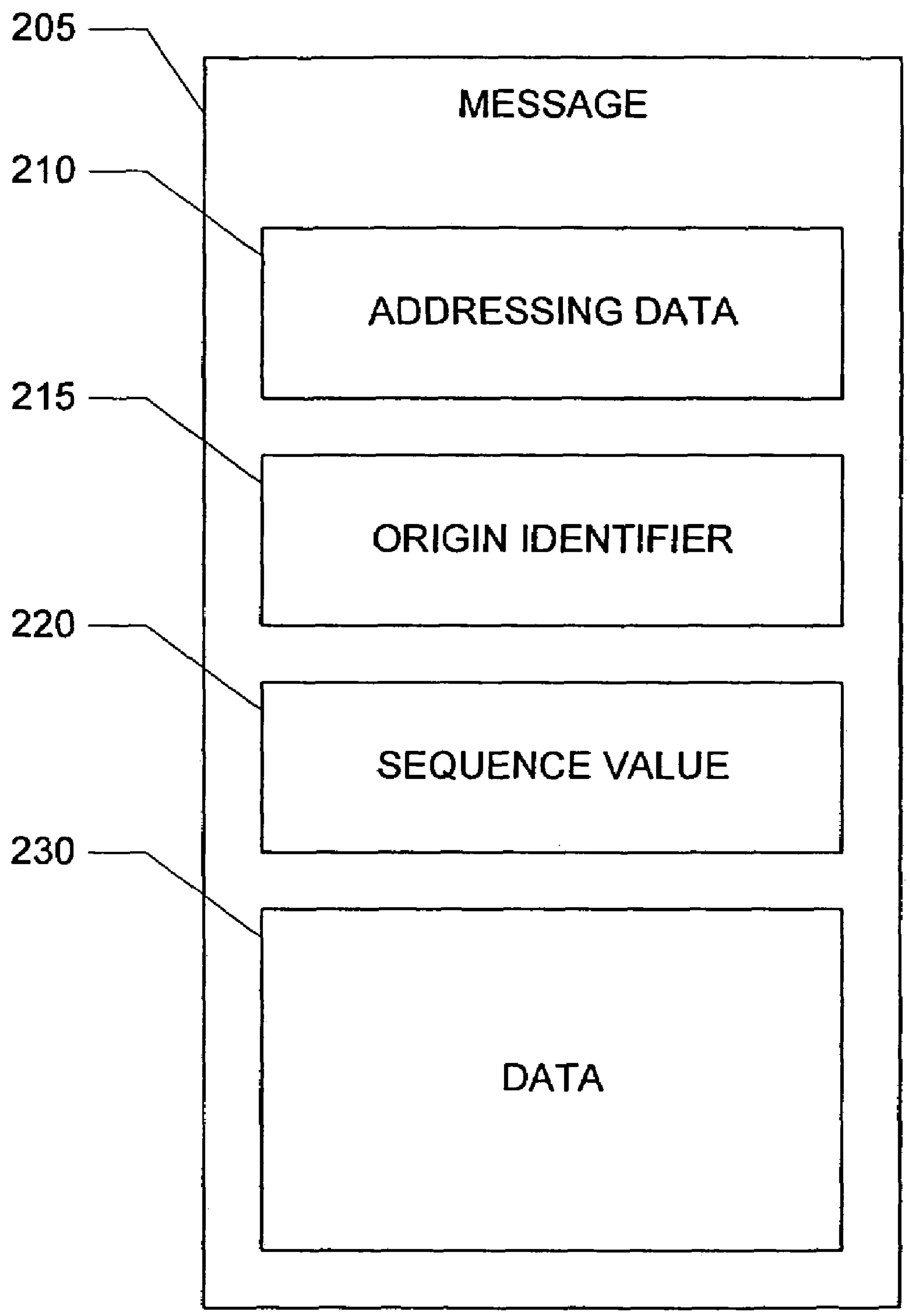

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

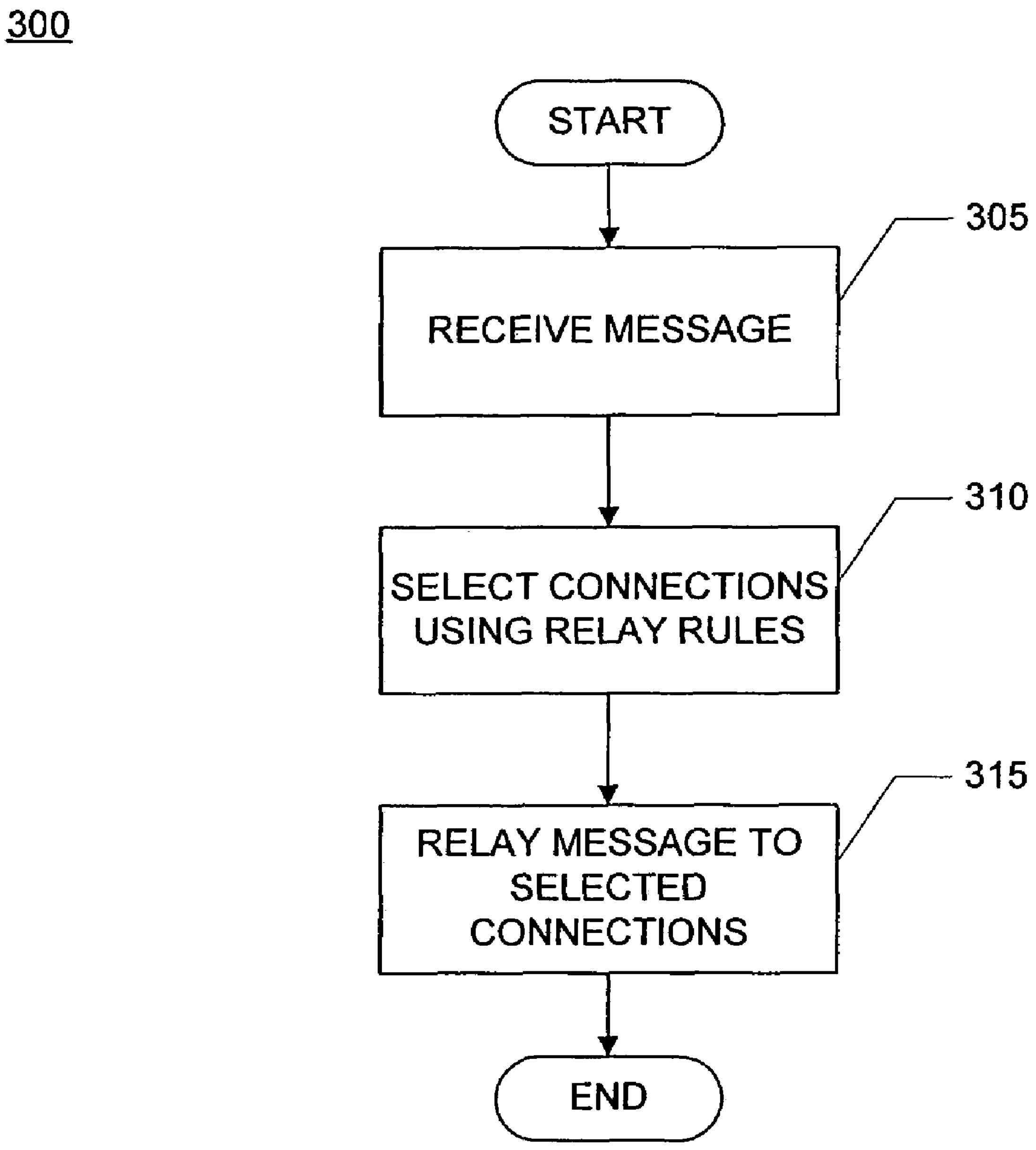

300
START
305
RECEIVE MESSAGE
310
SELECT CONNECTIONS USING RELAY RULES
315
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 4

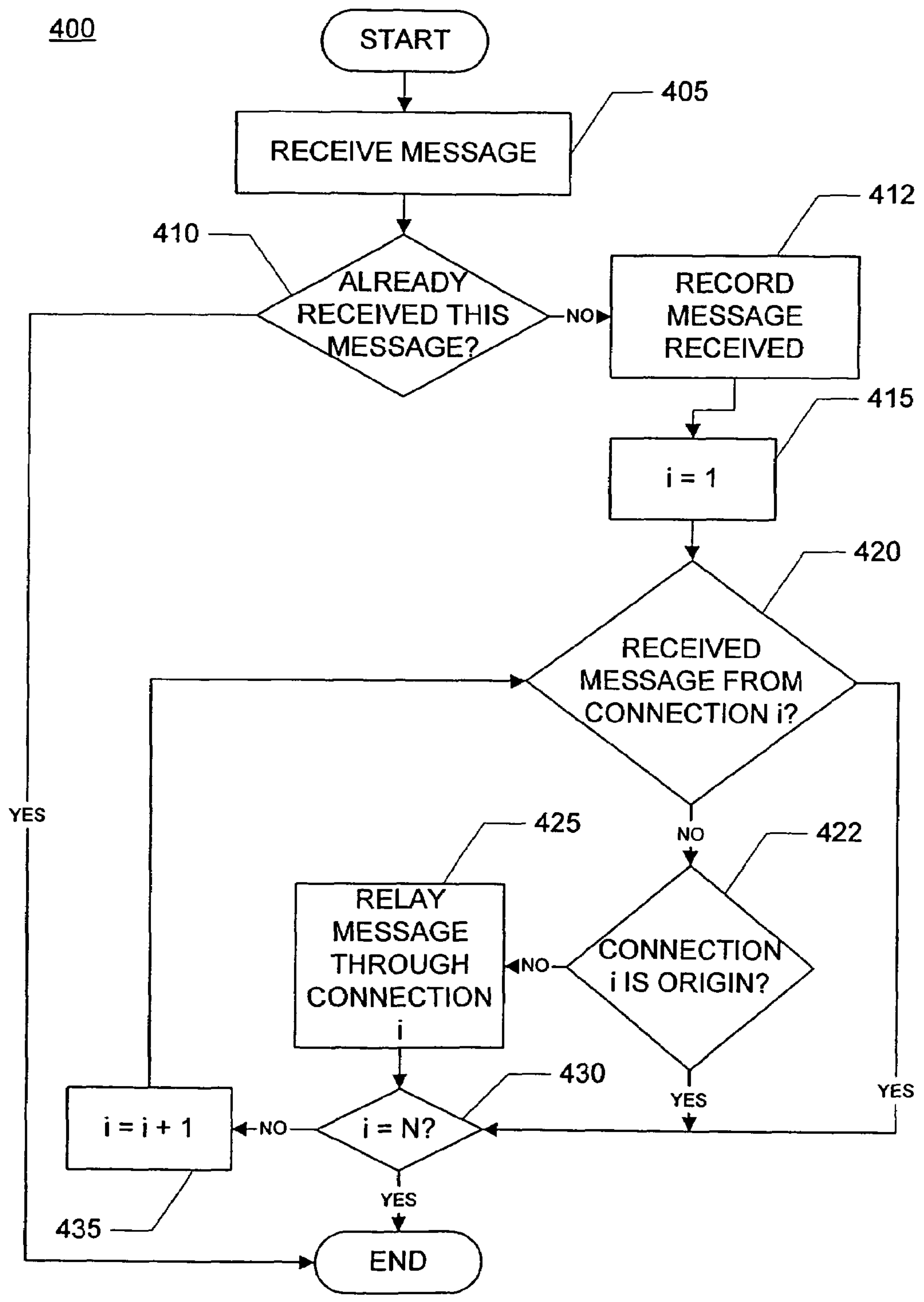

400
START
405
RECEIVE MESSAGE
410
ALREADY RECEIVED THIS MESSAGE?
NO
412
RECORD MESSAGE RECEIVED
415
i = 1
420
RECEIVED MESSAGE FROM CONNECTION i?
YES
NO
422
CONNECTION i IS ORIGIN?
NO
425
RELAY MESSAGE THROUGH CONNECTION i
YES
YES
430
i = N?
NO
i = i + 1
435
YES
END

FIG. 5

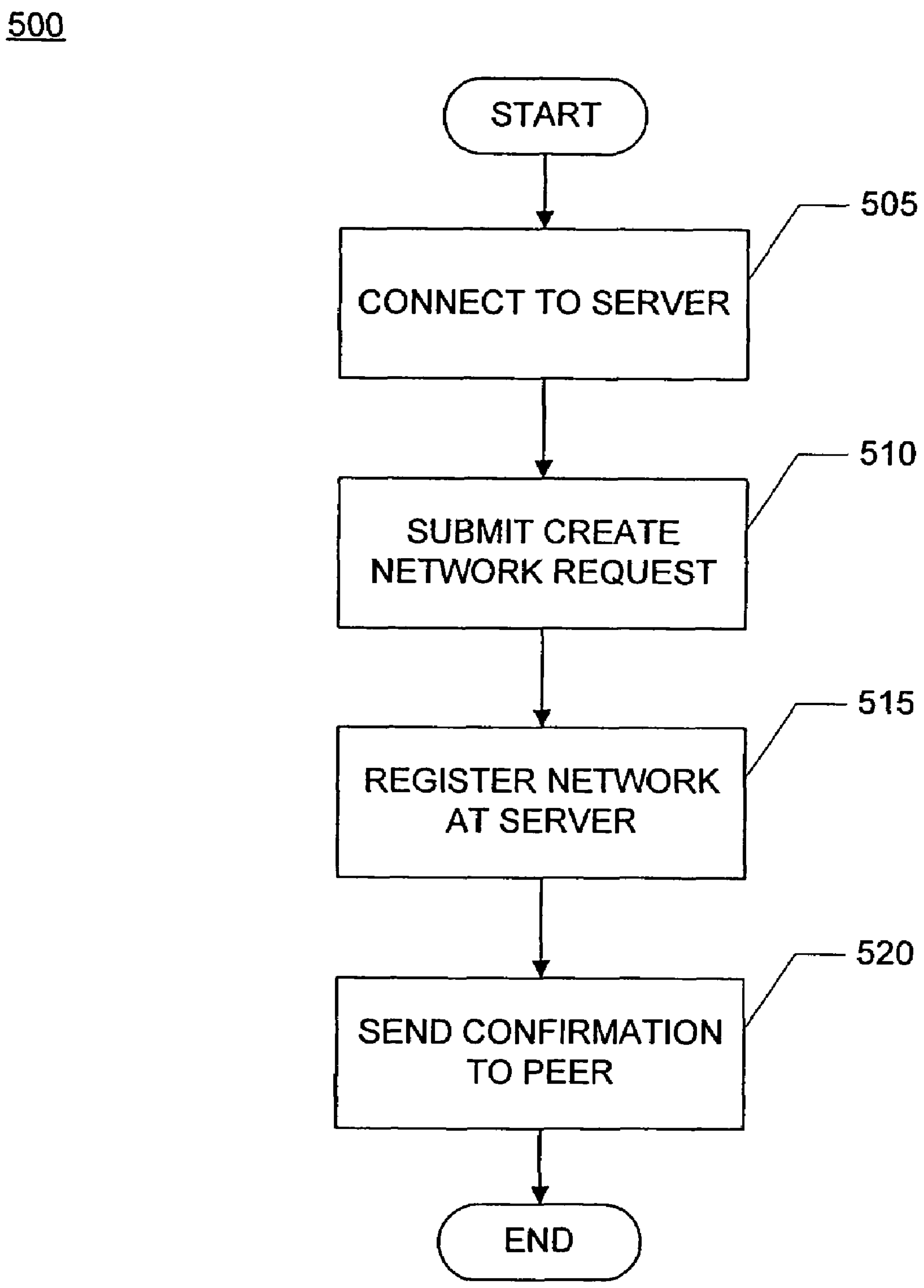

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

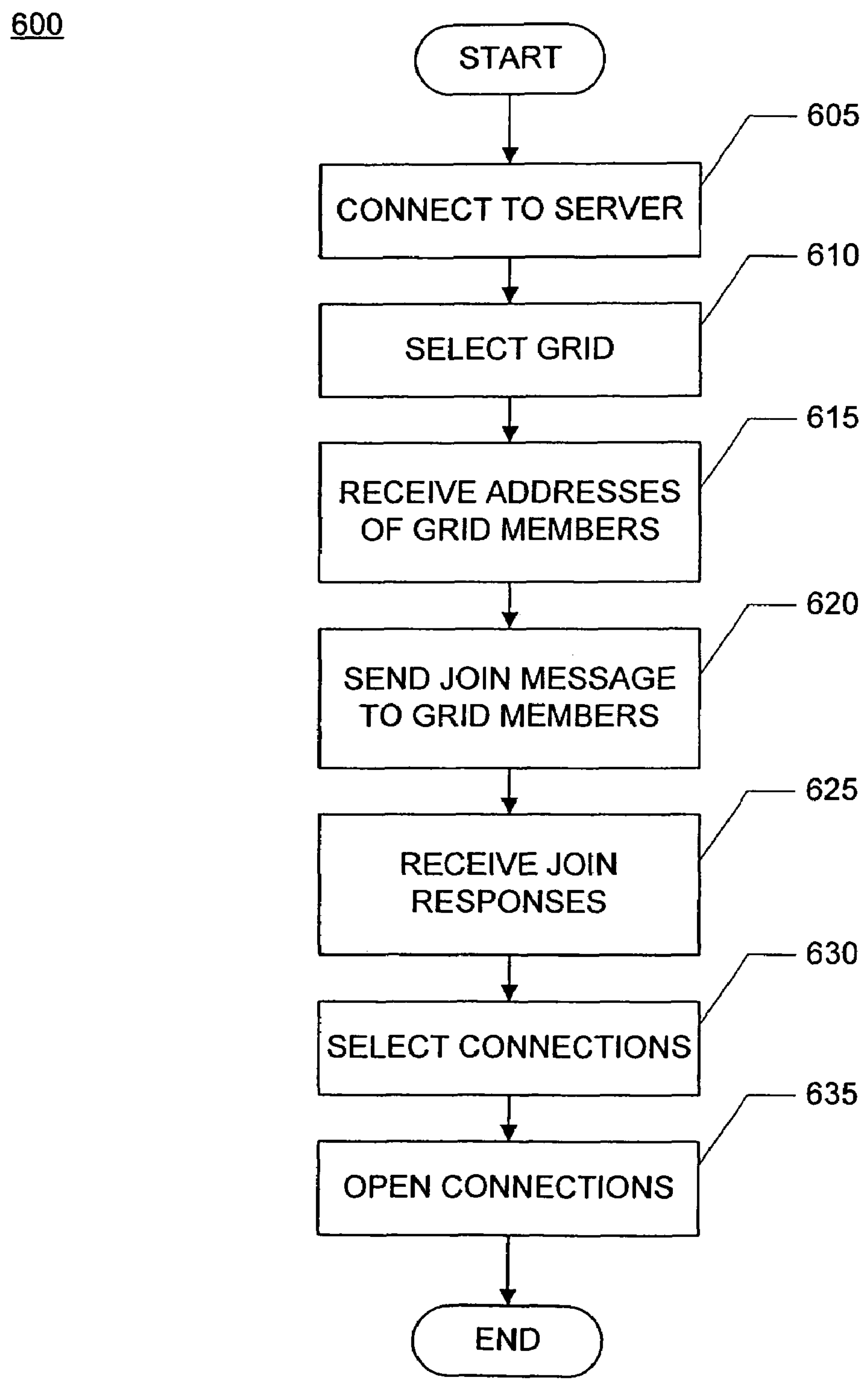

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

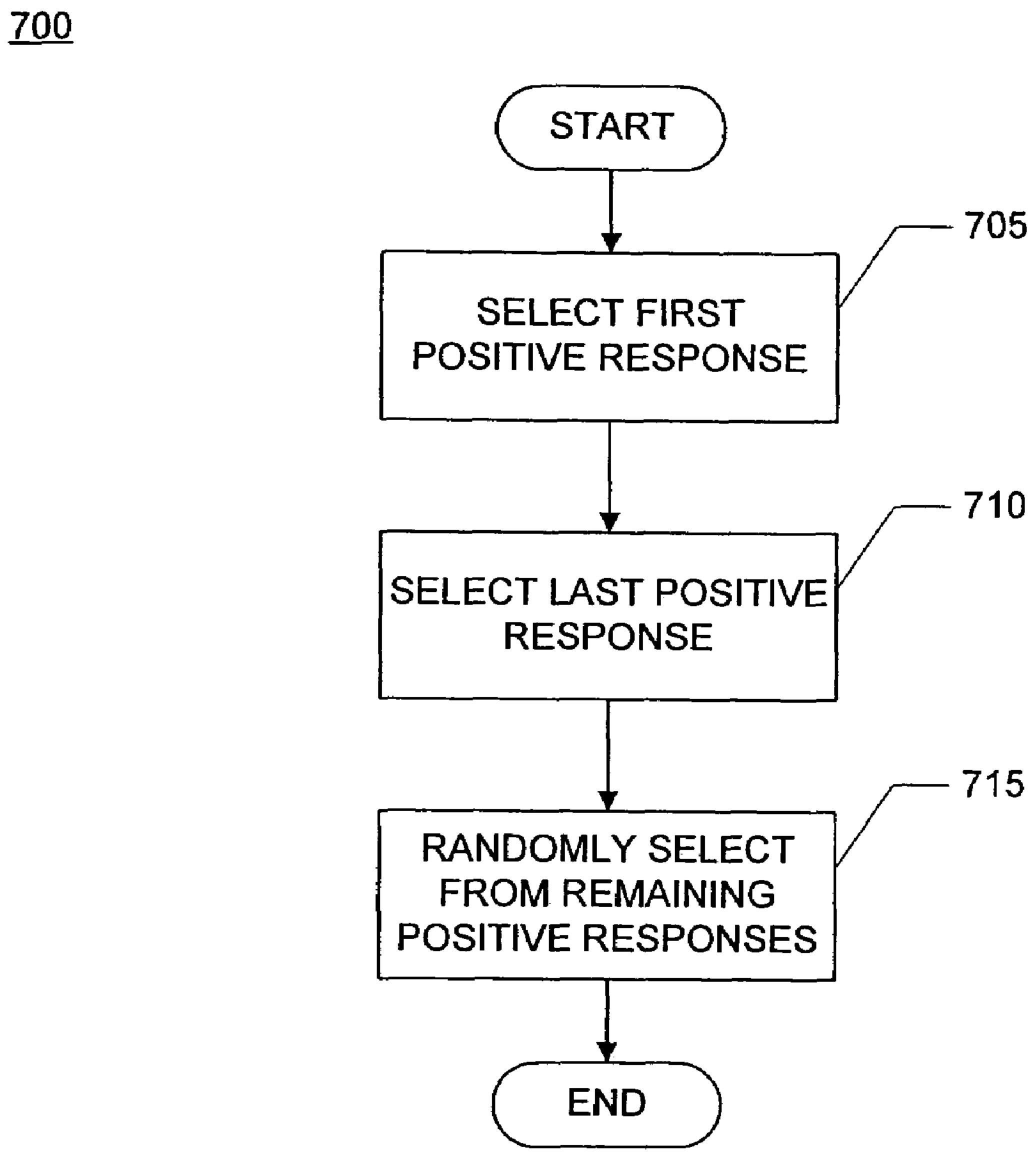

700
START
705
SELECT FIRST POSITIVE RESPONSE
710
SELECT LAST POSITIVE RESPONSE
715
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
END

FIG. 8

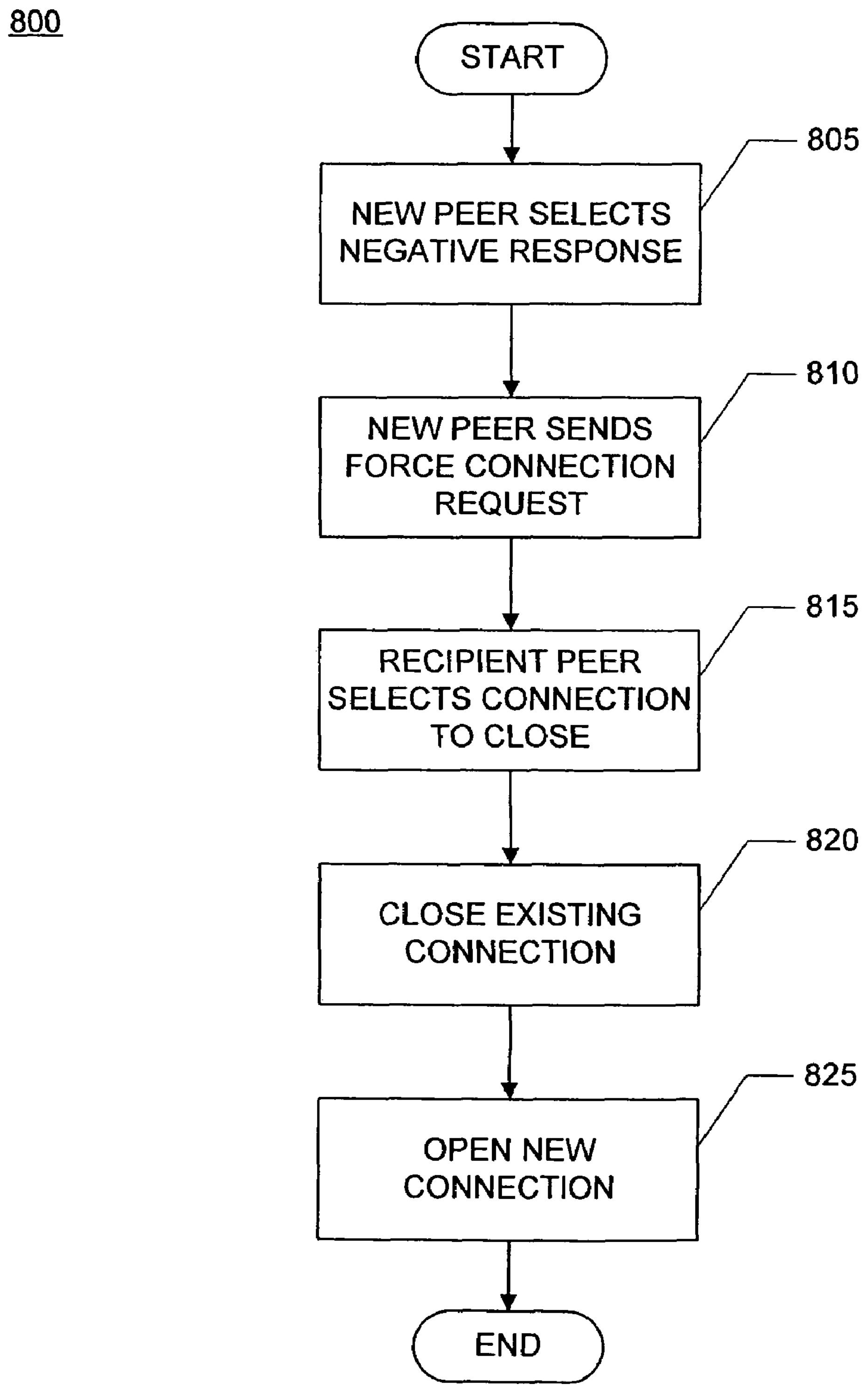

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

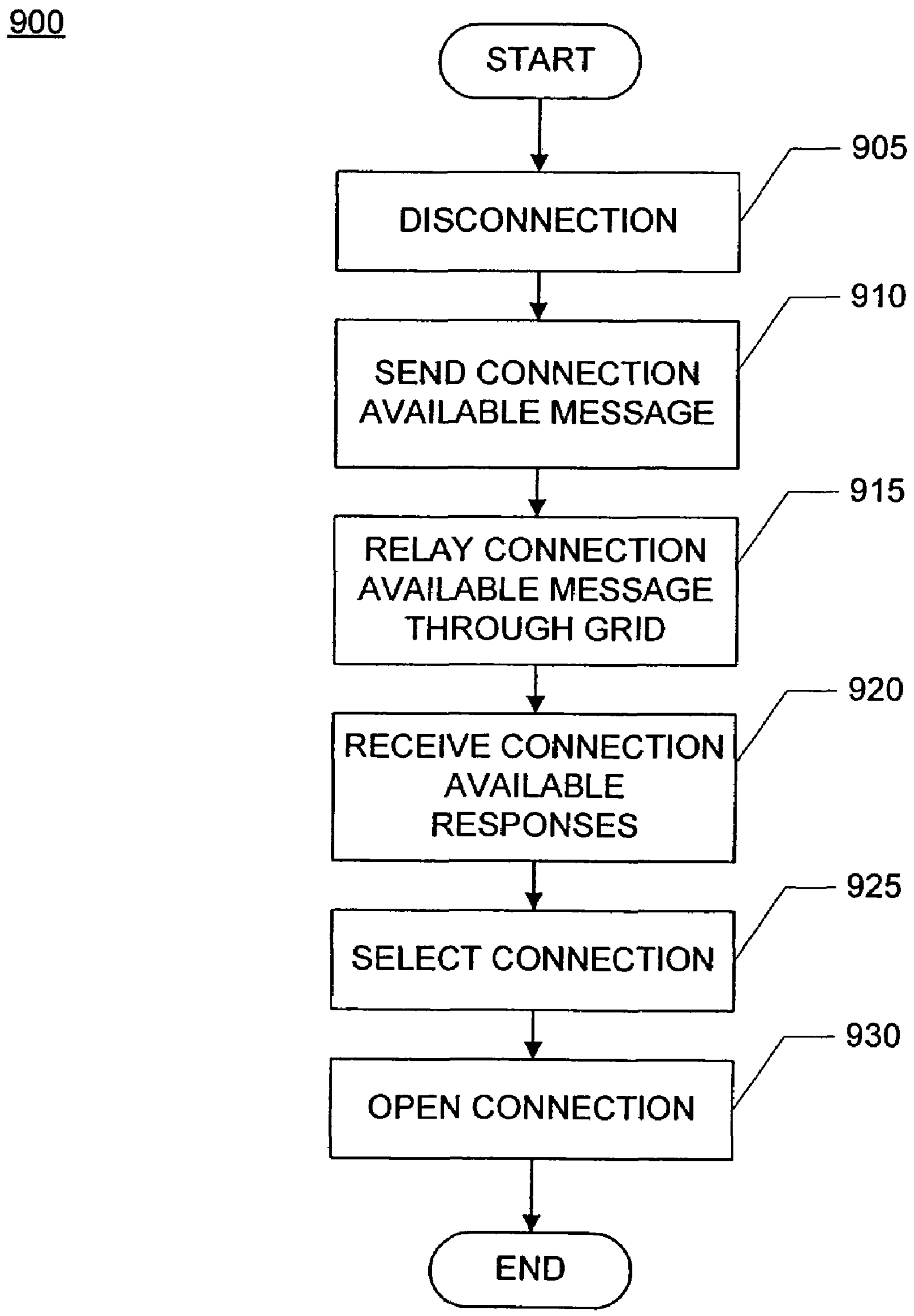

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

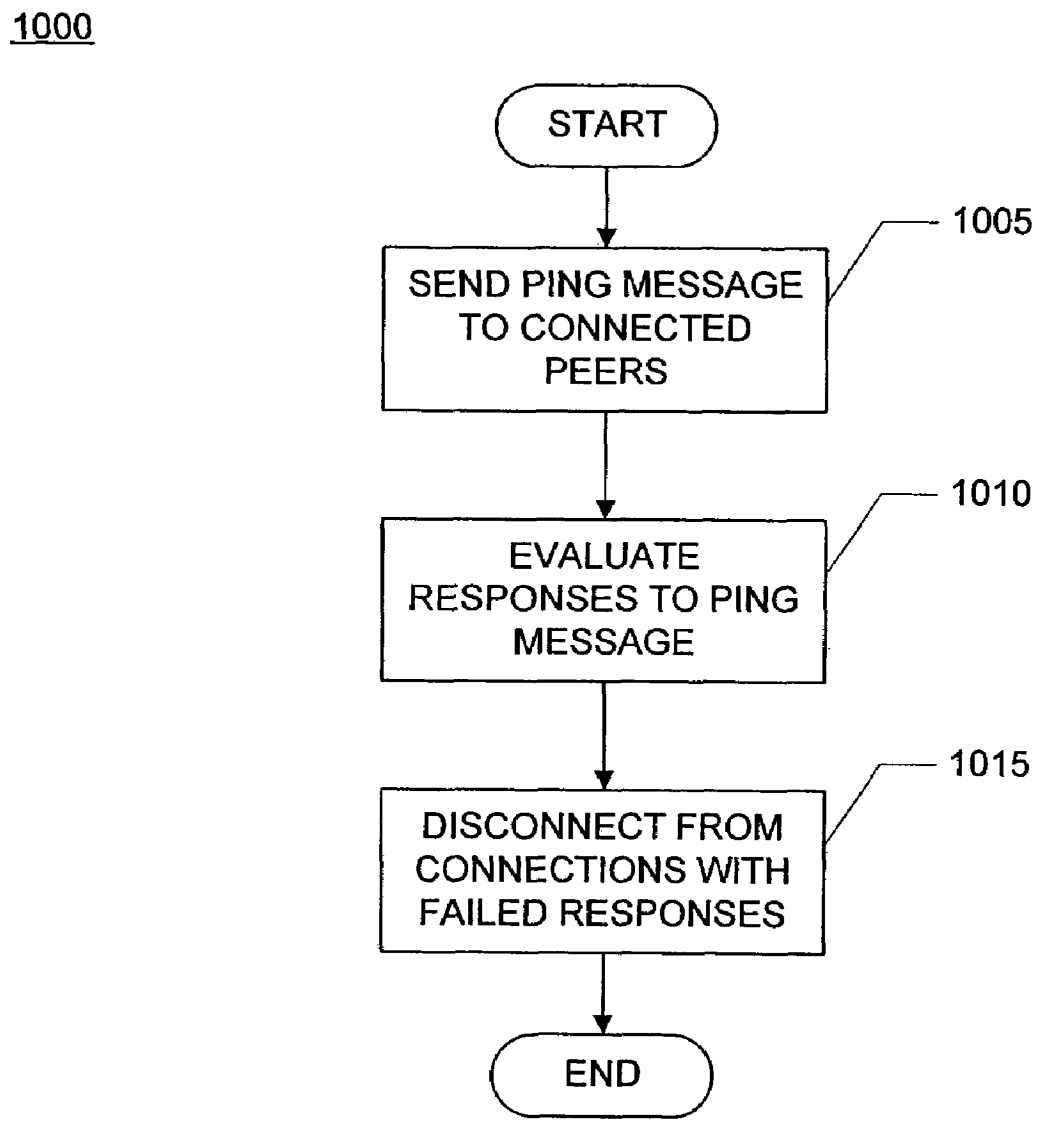

1000
START
1005
SEND PING MESSAGE TO CONNECTED PEERS
1010
EVALUATE RESPONSES TO PING MESSAGE
1015
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
END

FIG. 11

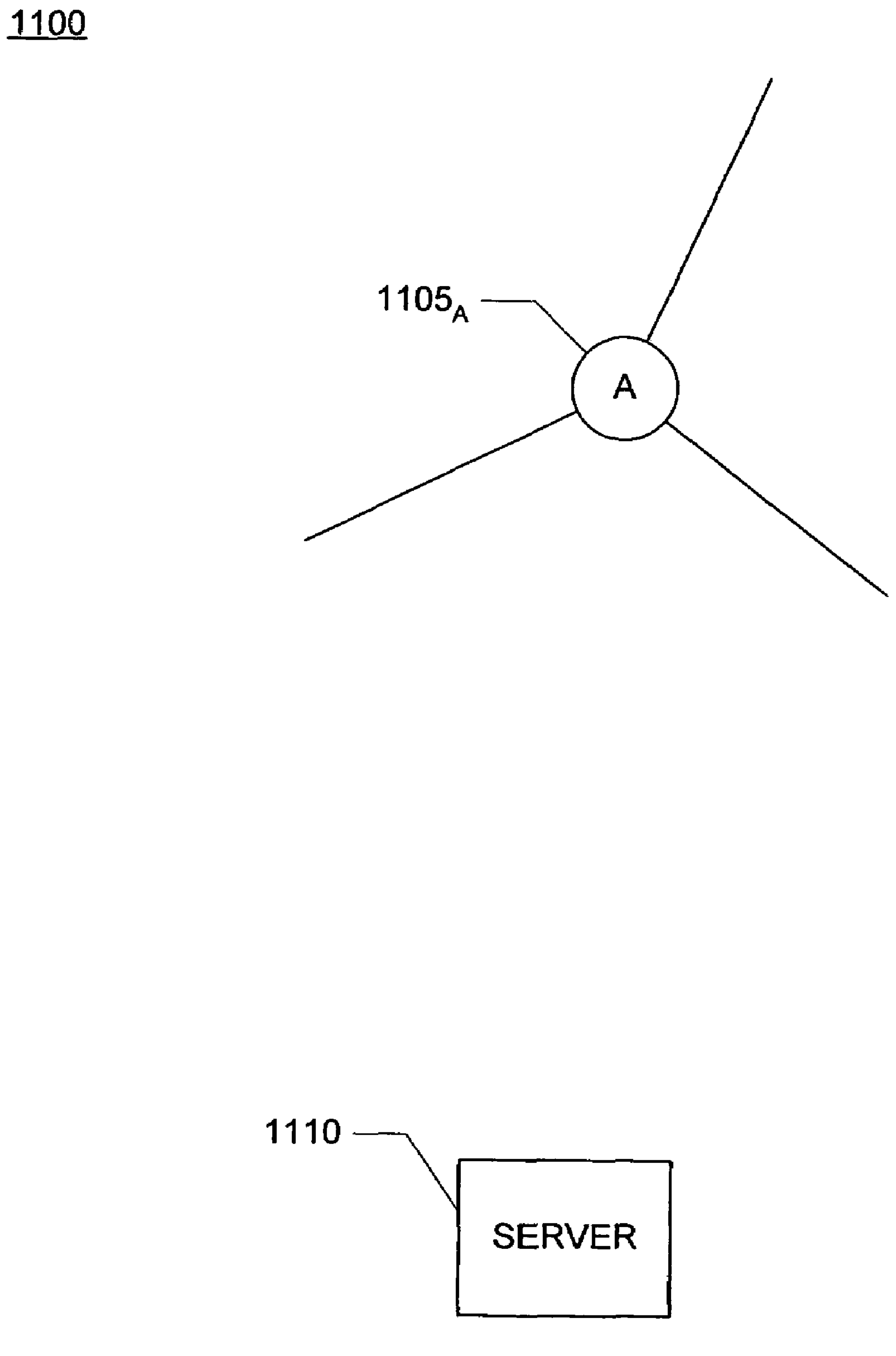


FIG. 12

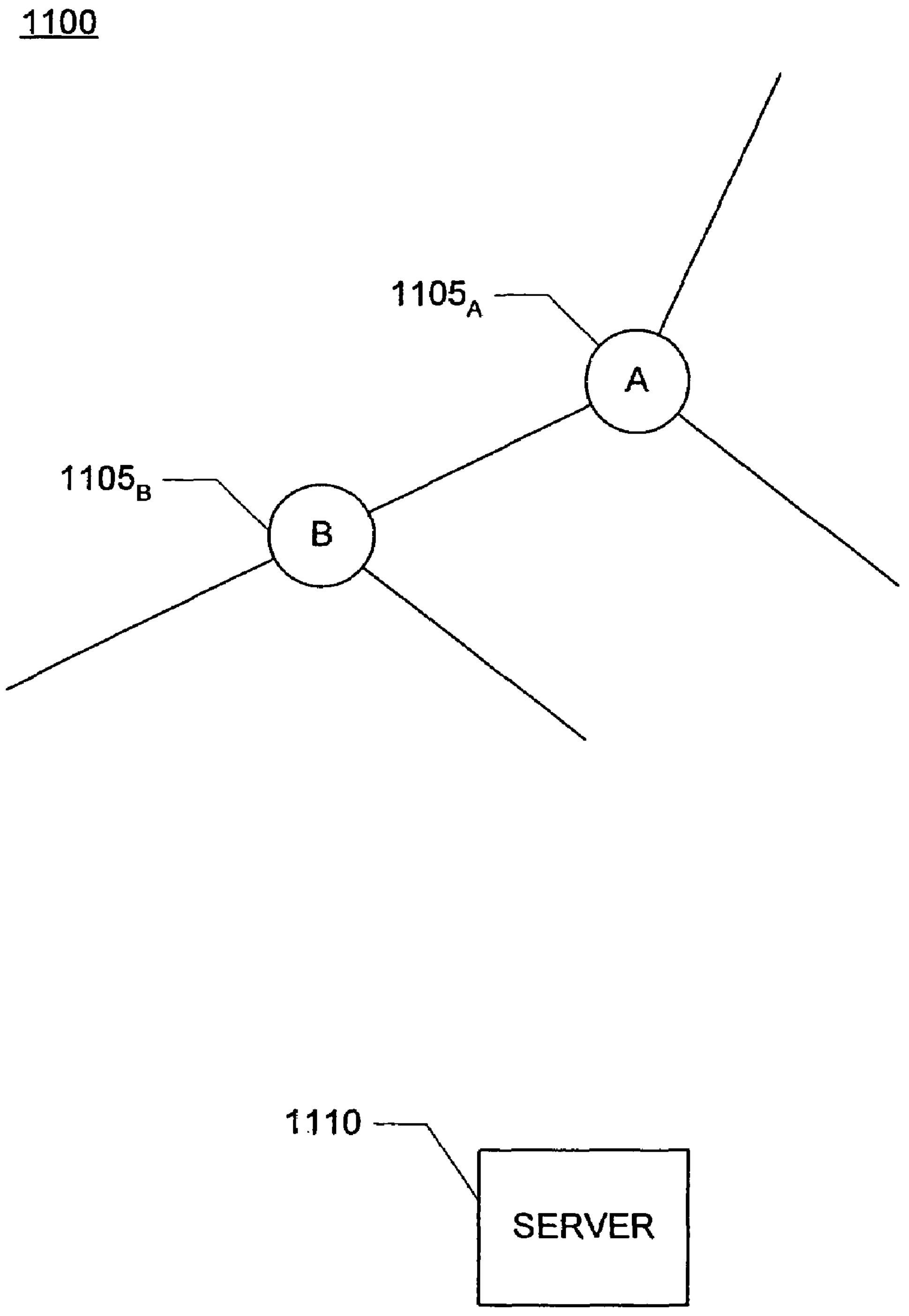


FIG. 13

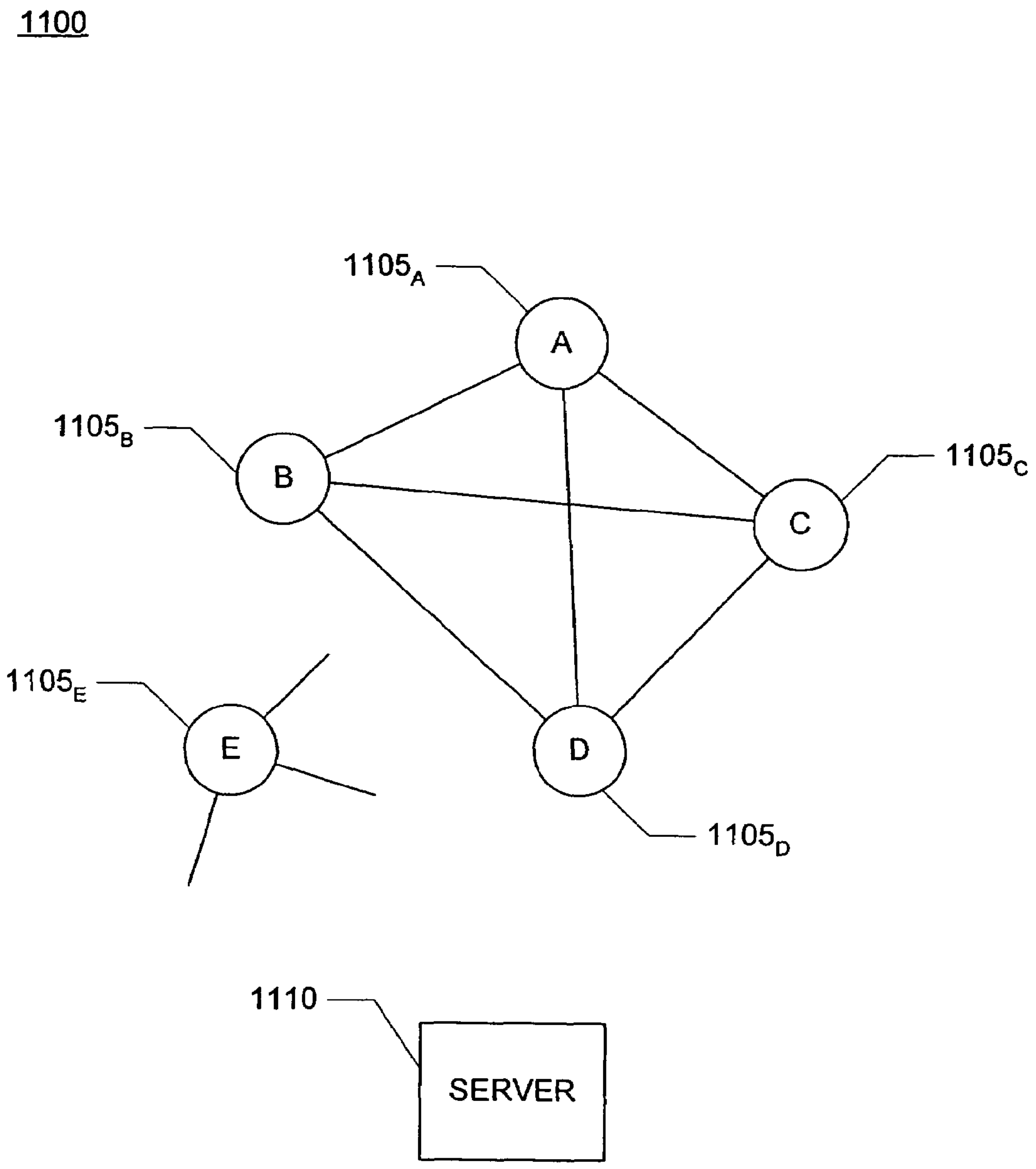


FIG. 14

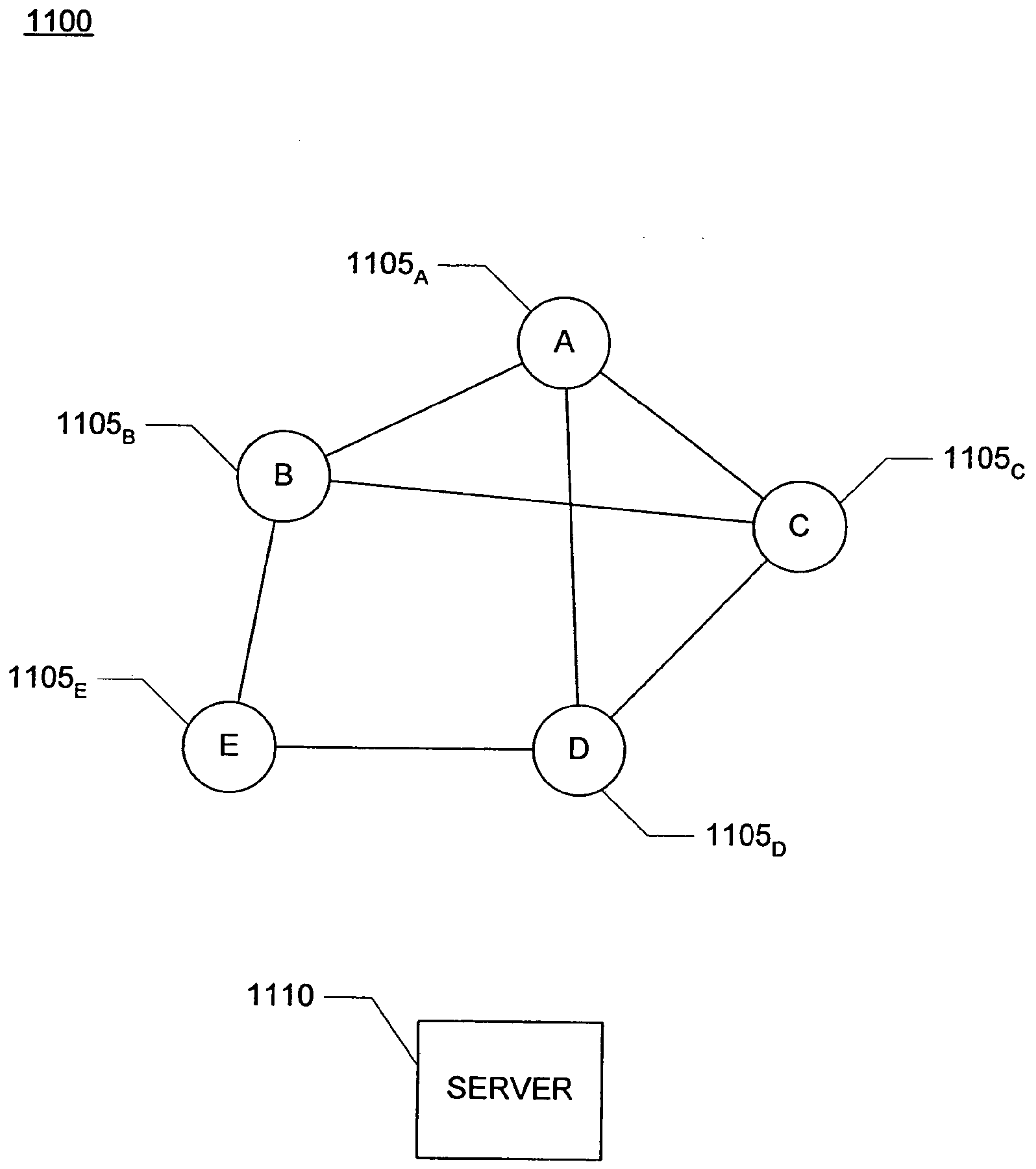


FIG. 15

1100

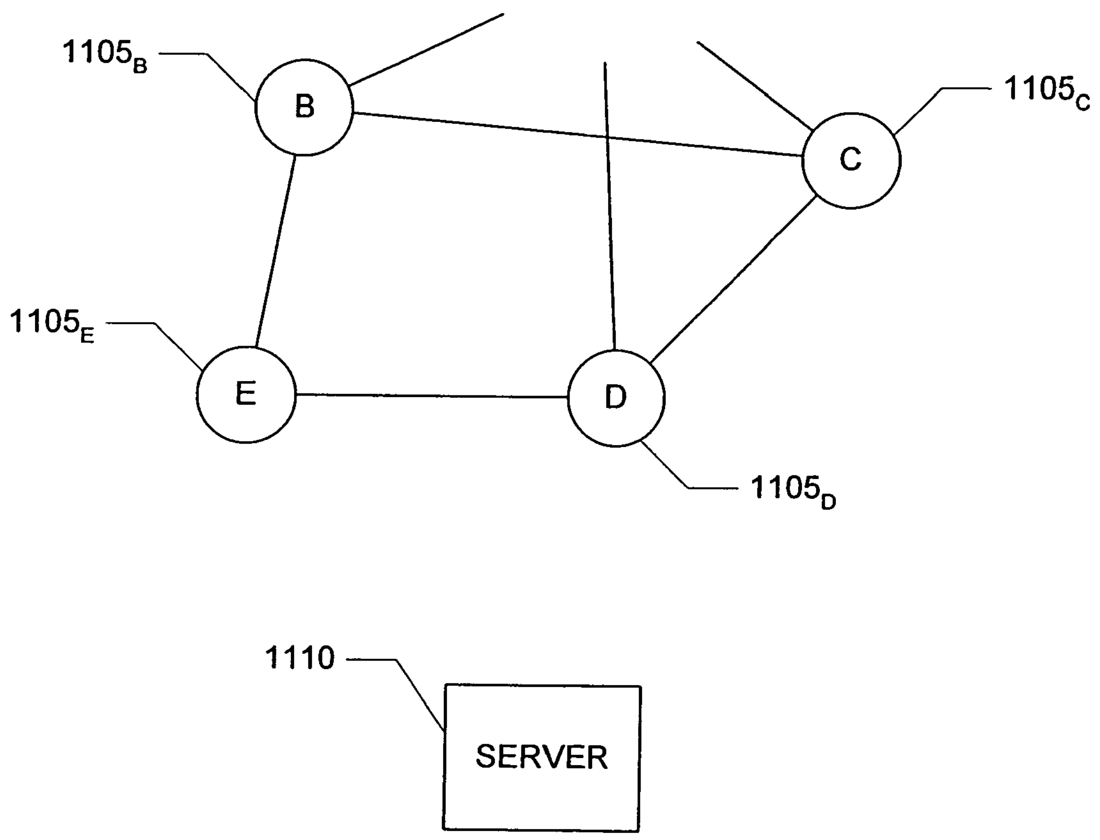


FIG. 16

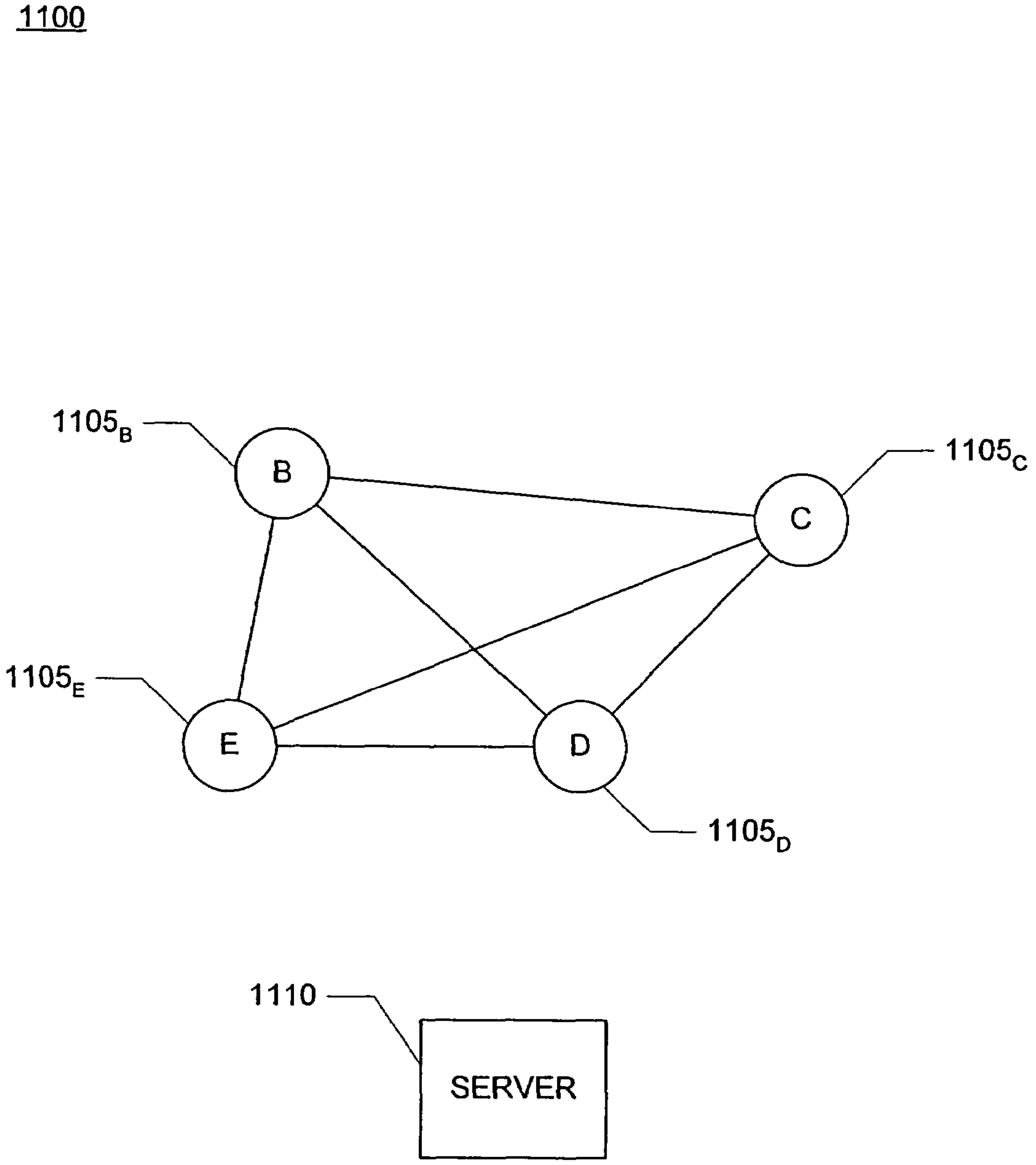


FIG. 17

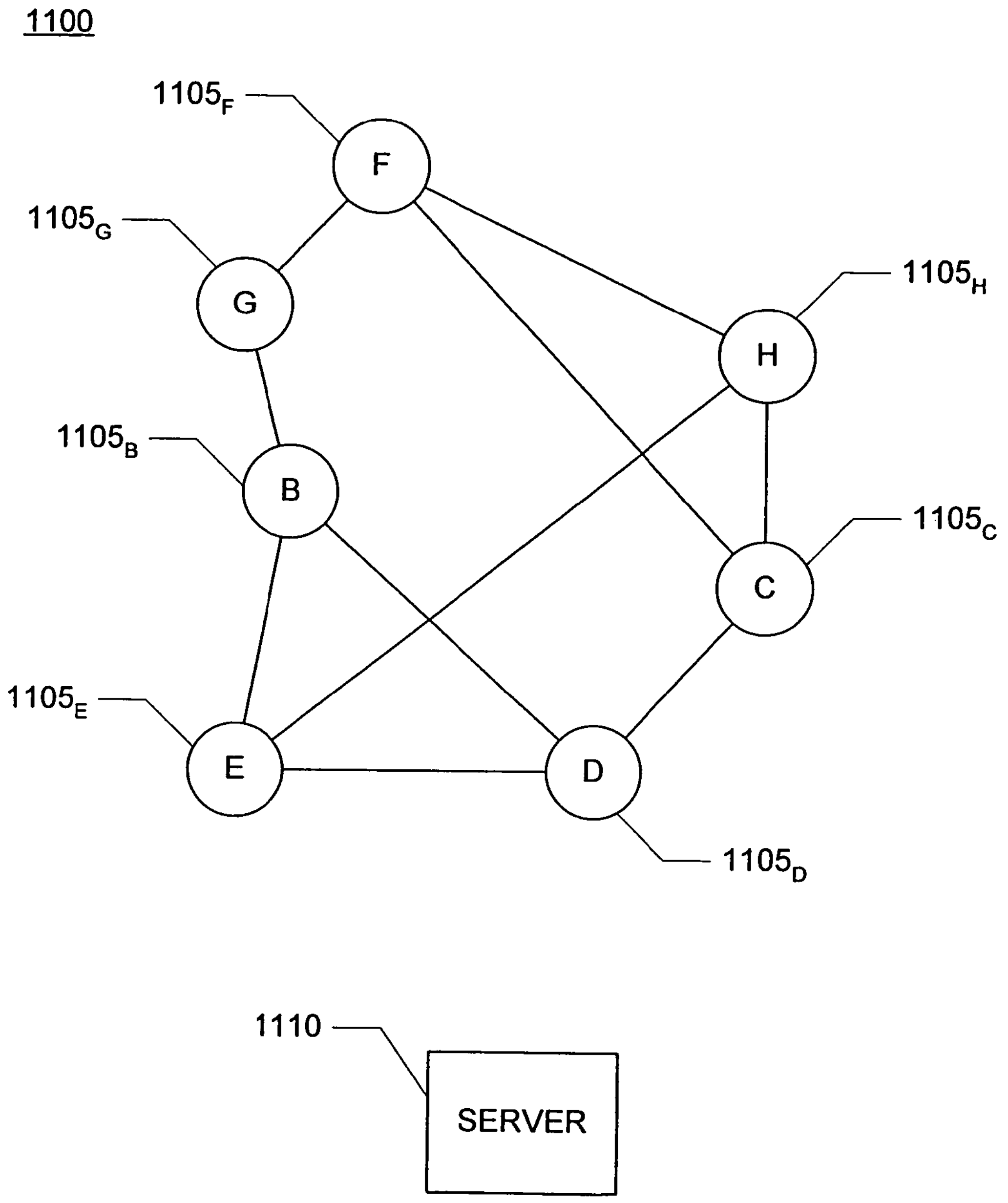

1100
1105$_F$
F
1105$_G$
G
1105$_H$
H
1105$_B$
B
1105$_C$
C
1105$_E$
E
D
1105$_D$
1110
SERVER

FIG. 18

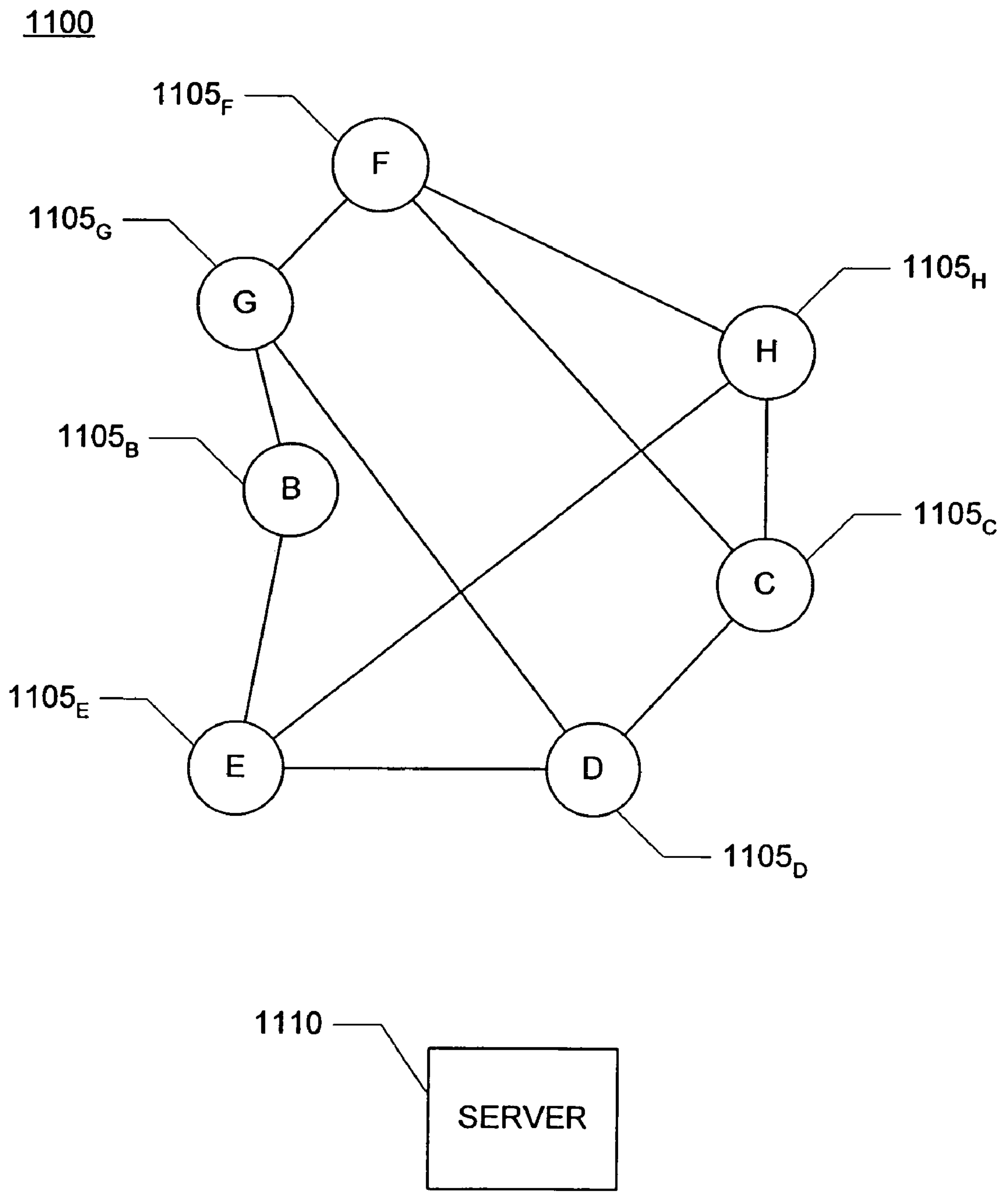

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 19

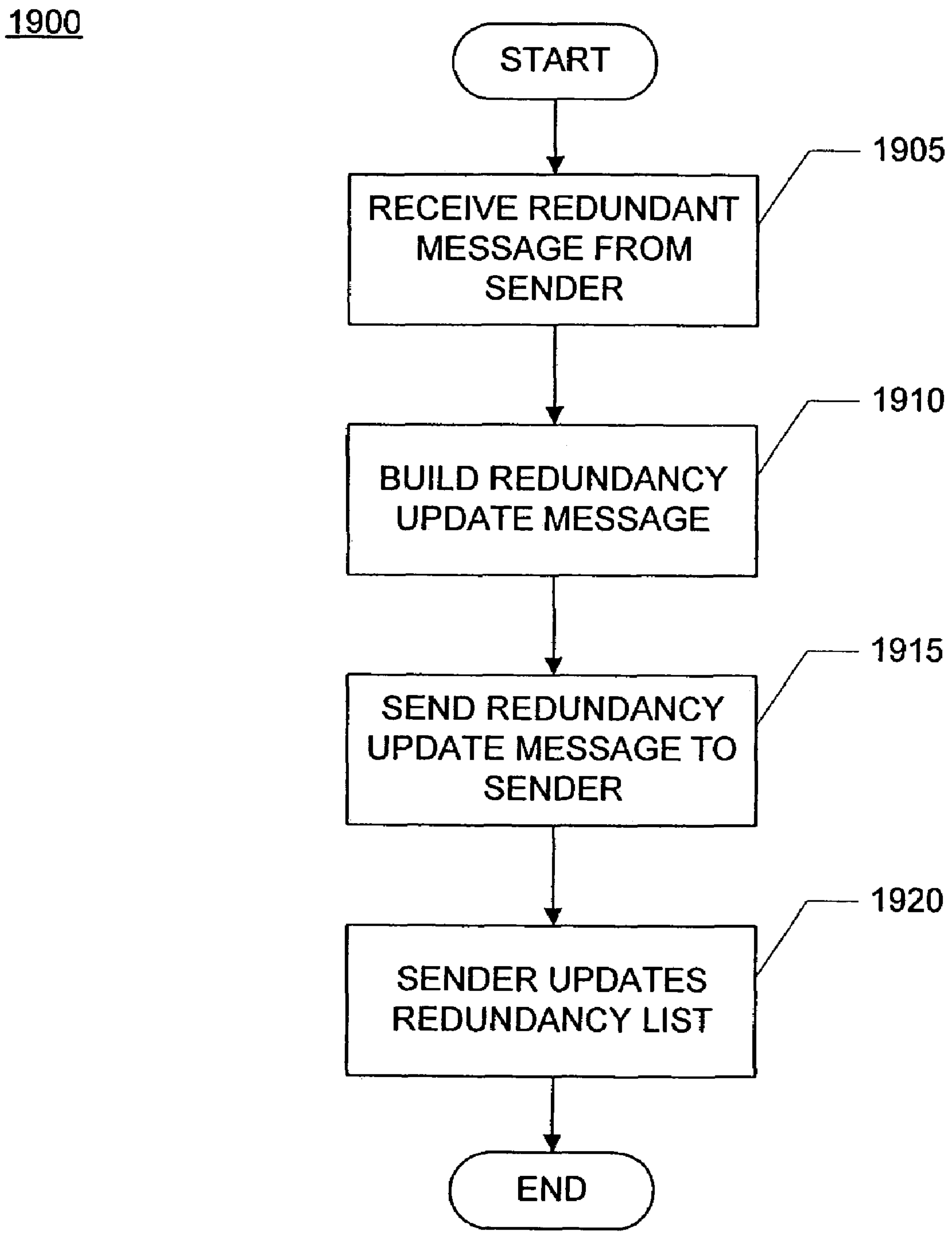

1900
START
1905
RECEIVE REDUNDANT MESSAGE FROM SENDER
1910
BUILD REDUNDANCY UPDATE MESSAGE
1915
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1920
SENDER UPDATES REDUNDANCY LIST
END

FIG. 20

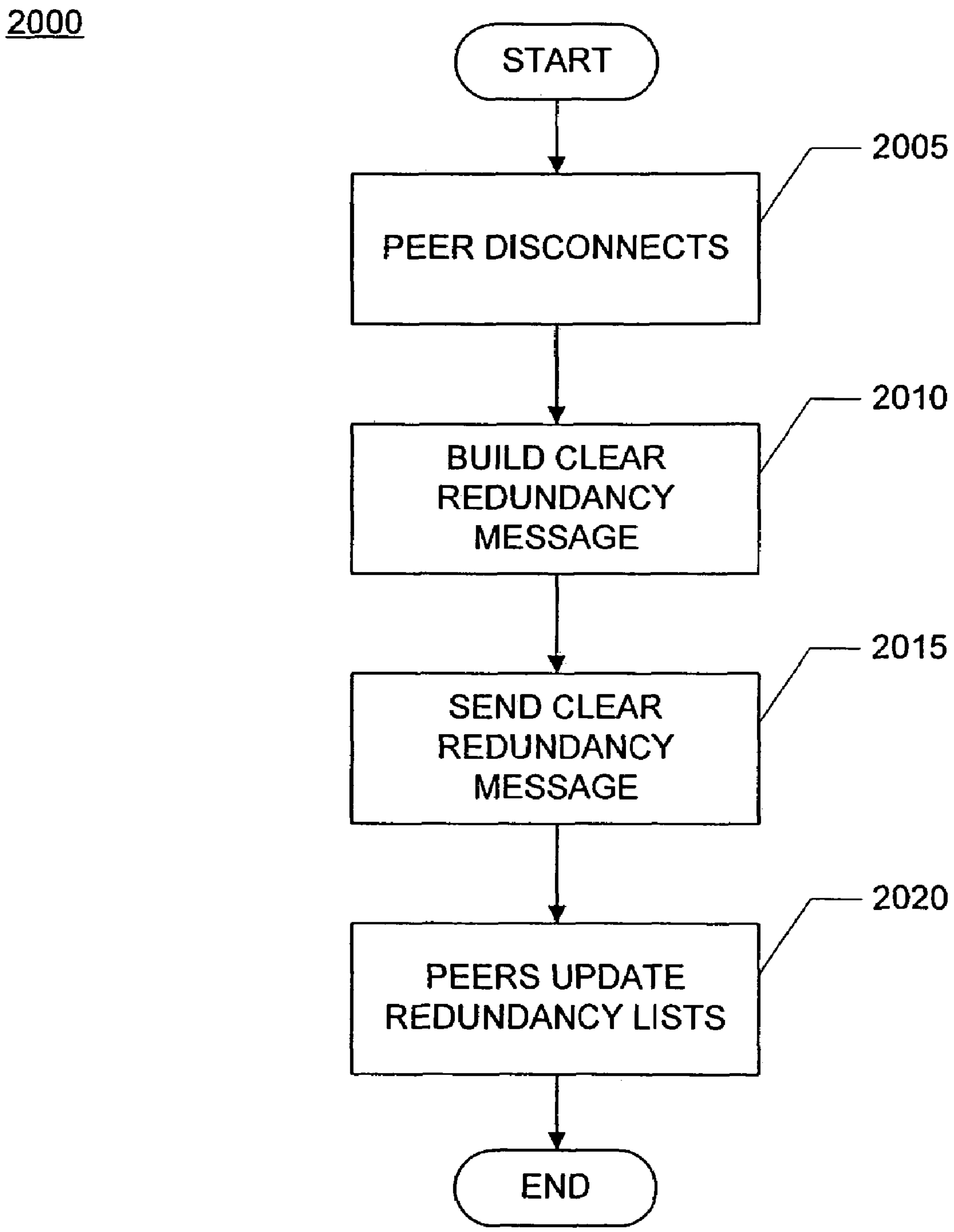

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

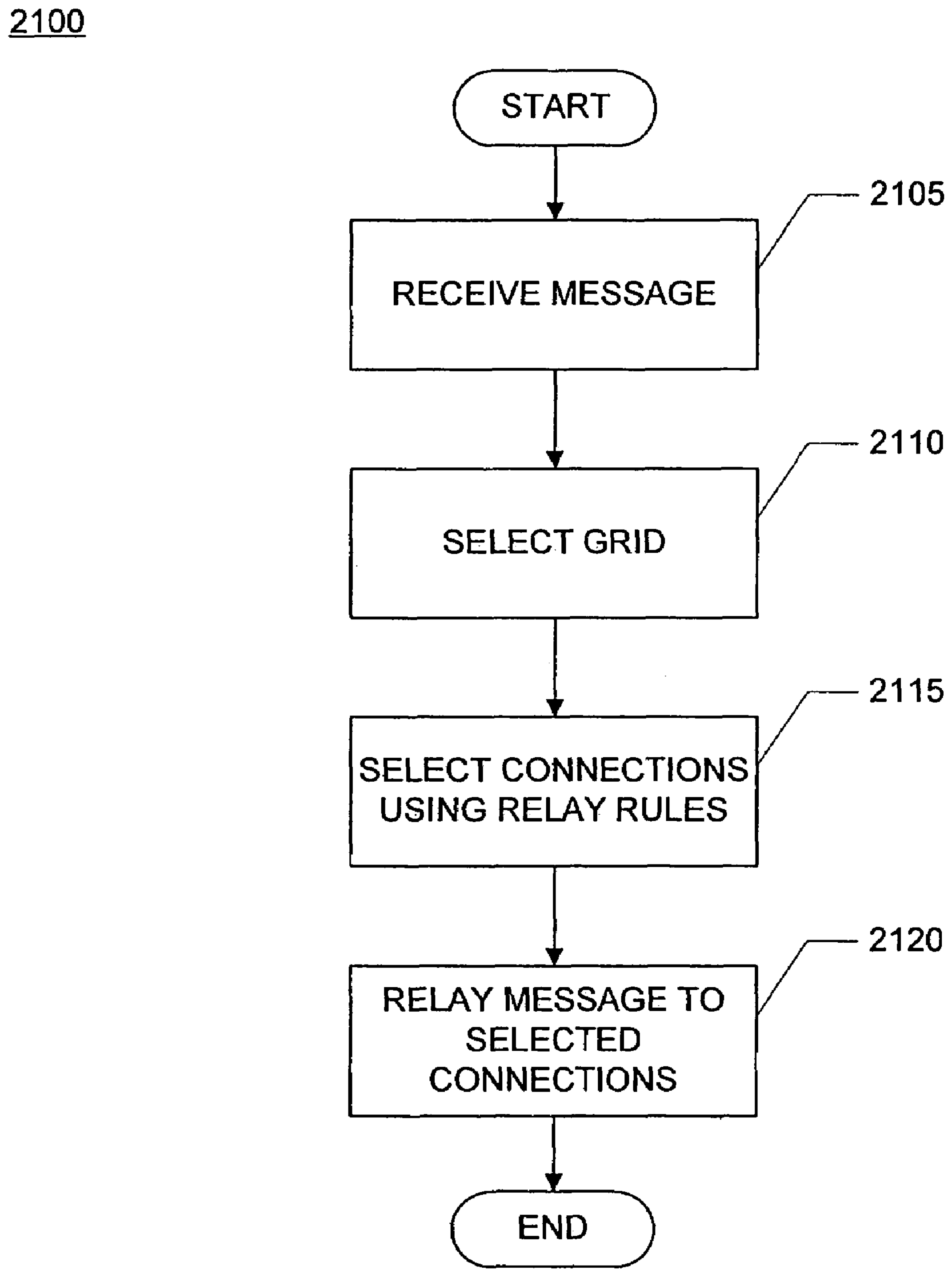

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

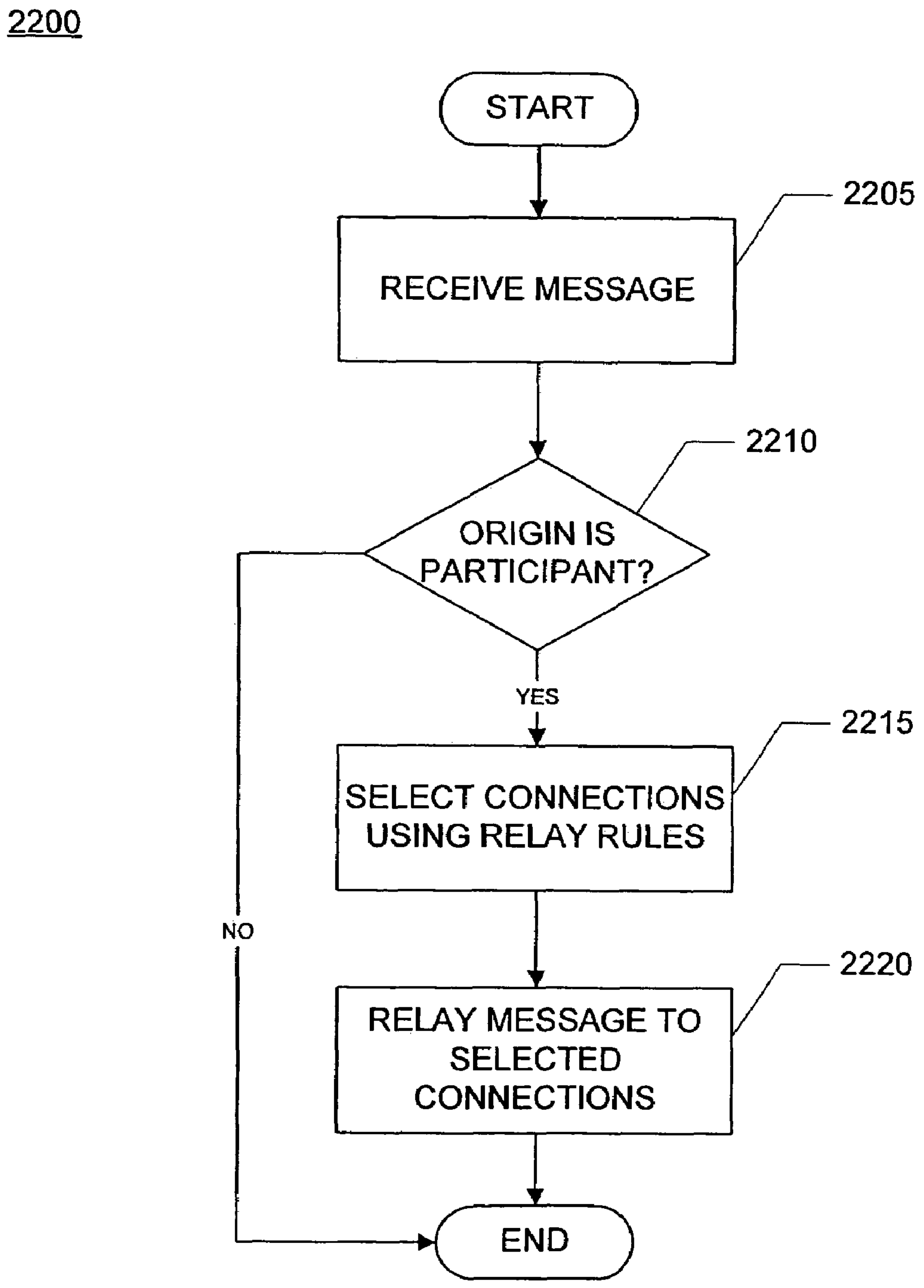

2200
START
2205
RECEIVE MESSAGE
2210
ORIGIN IS PARTICIPANT?
YES
NO
2215
SELECT CONNECTIONS USING RELAY RULES
2220
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 23

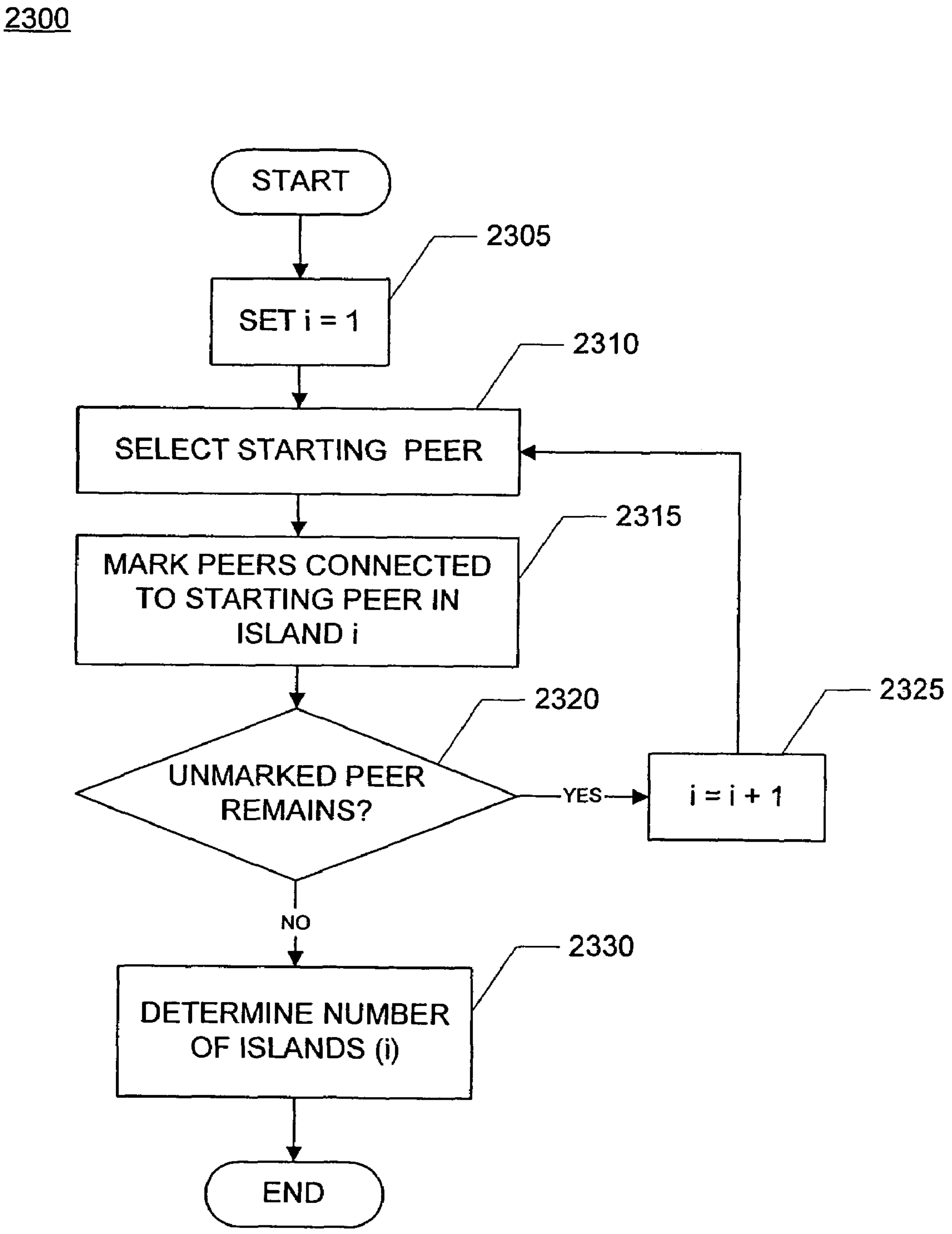

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

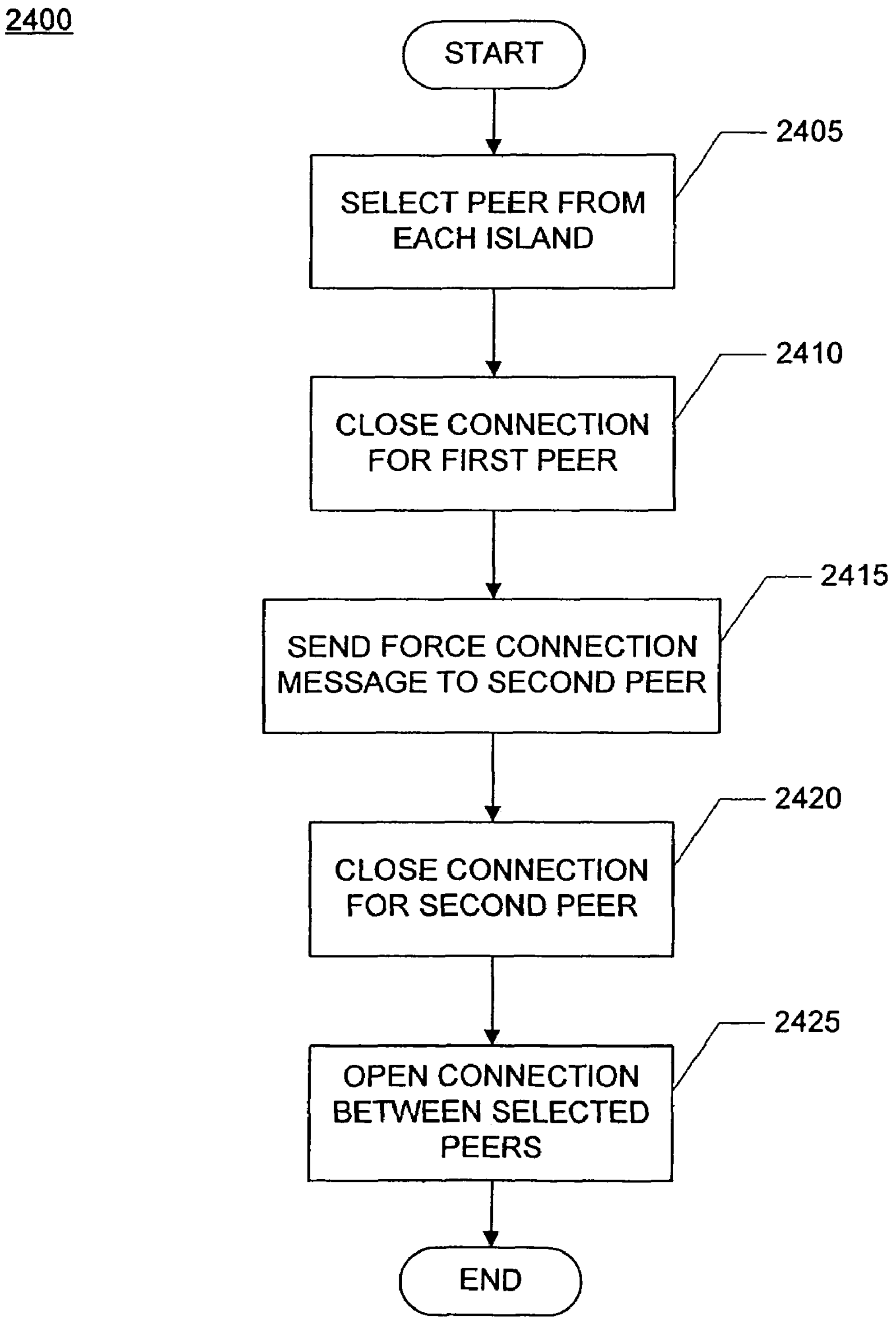

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

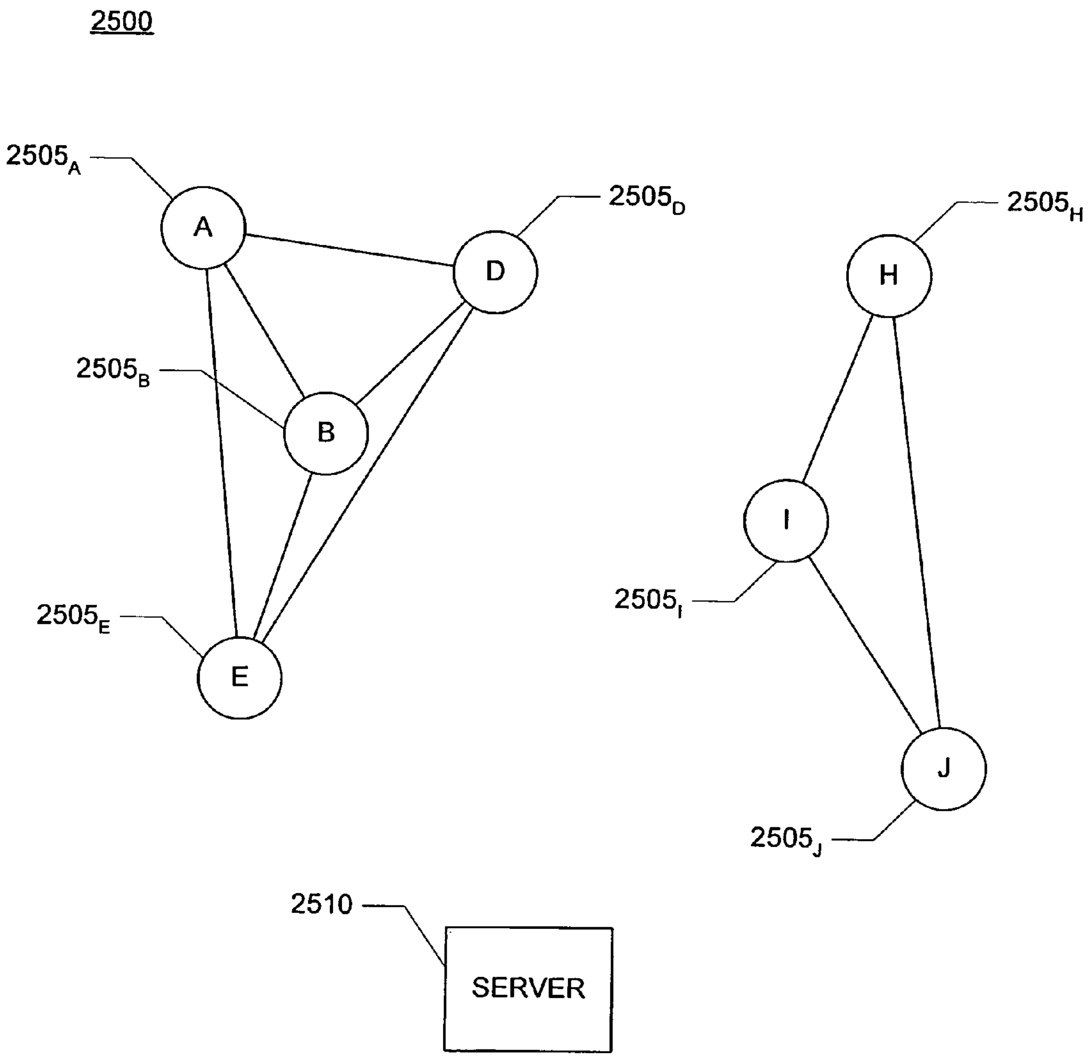

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 26

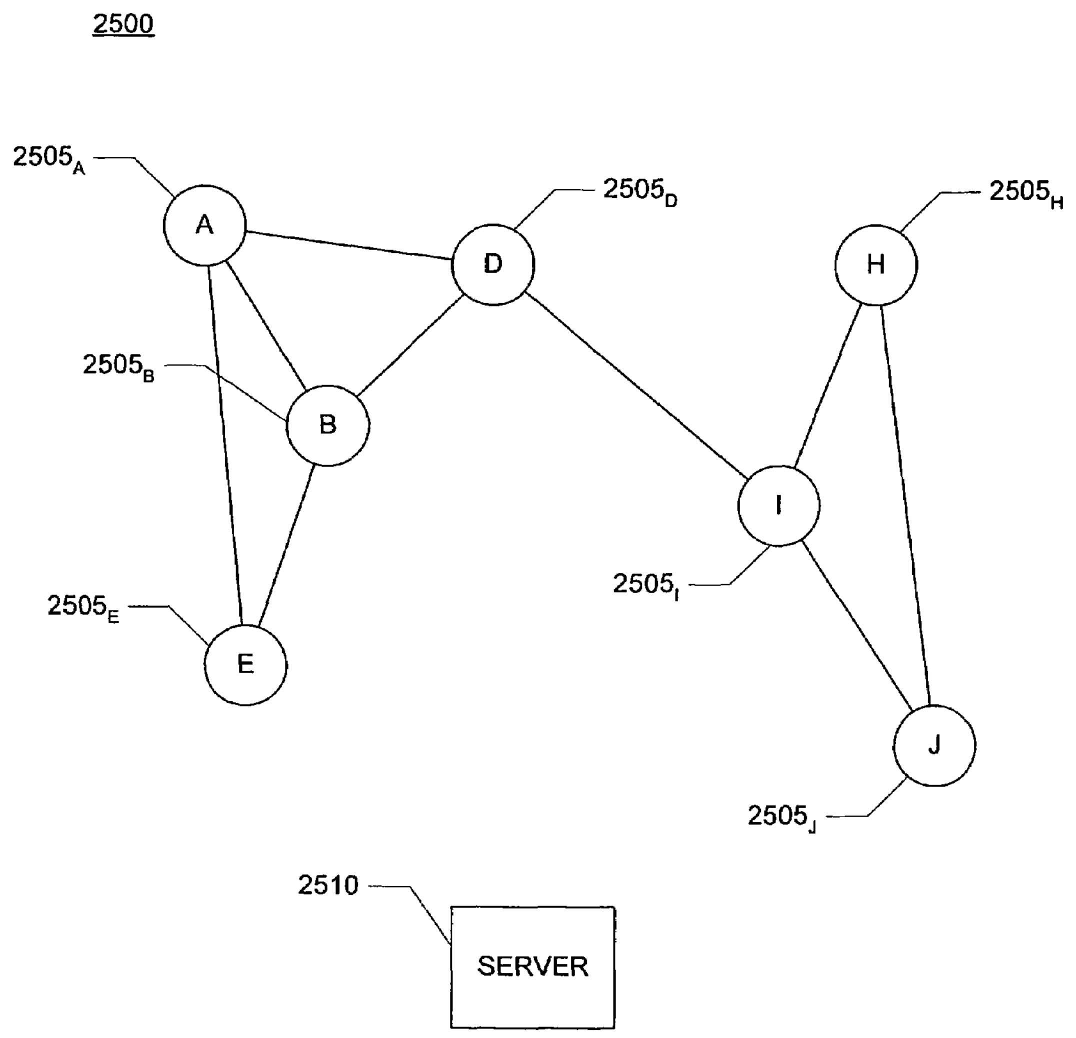

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 27

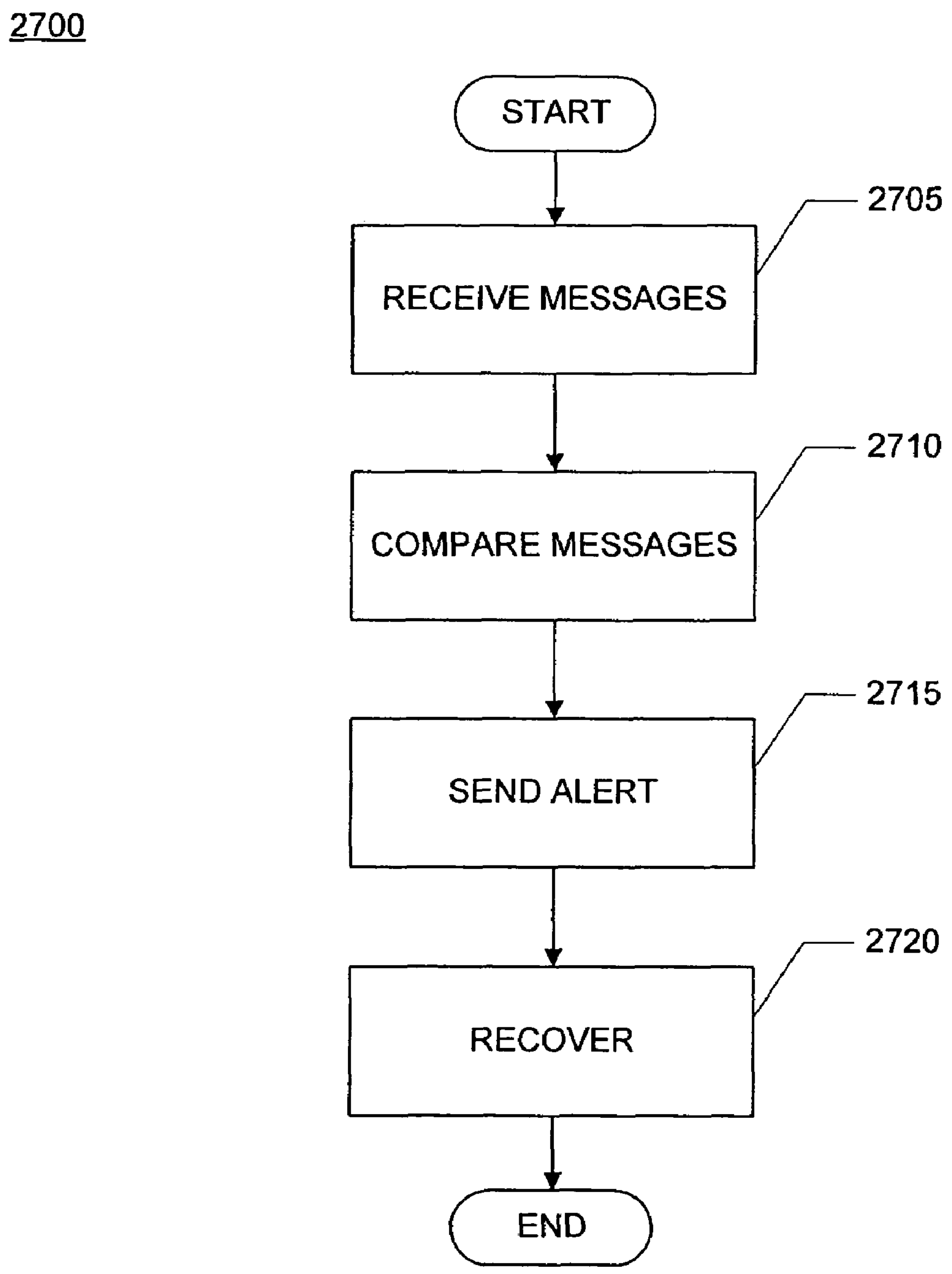

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

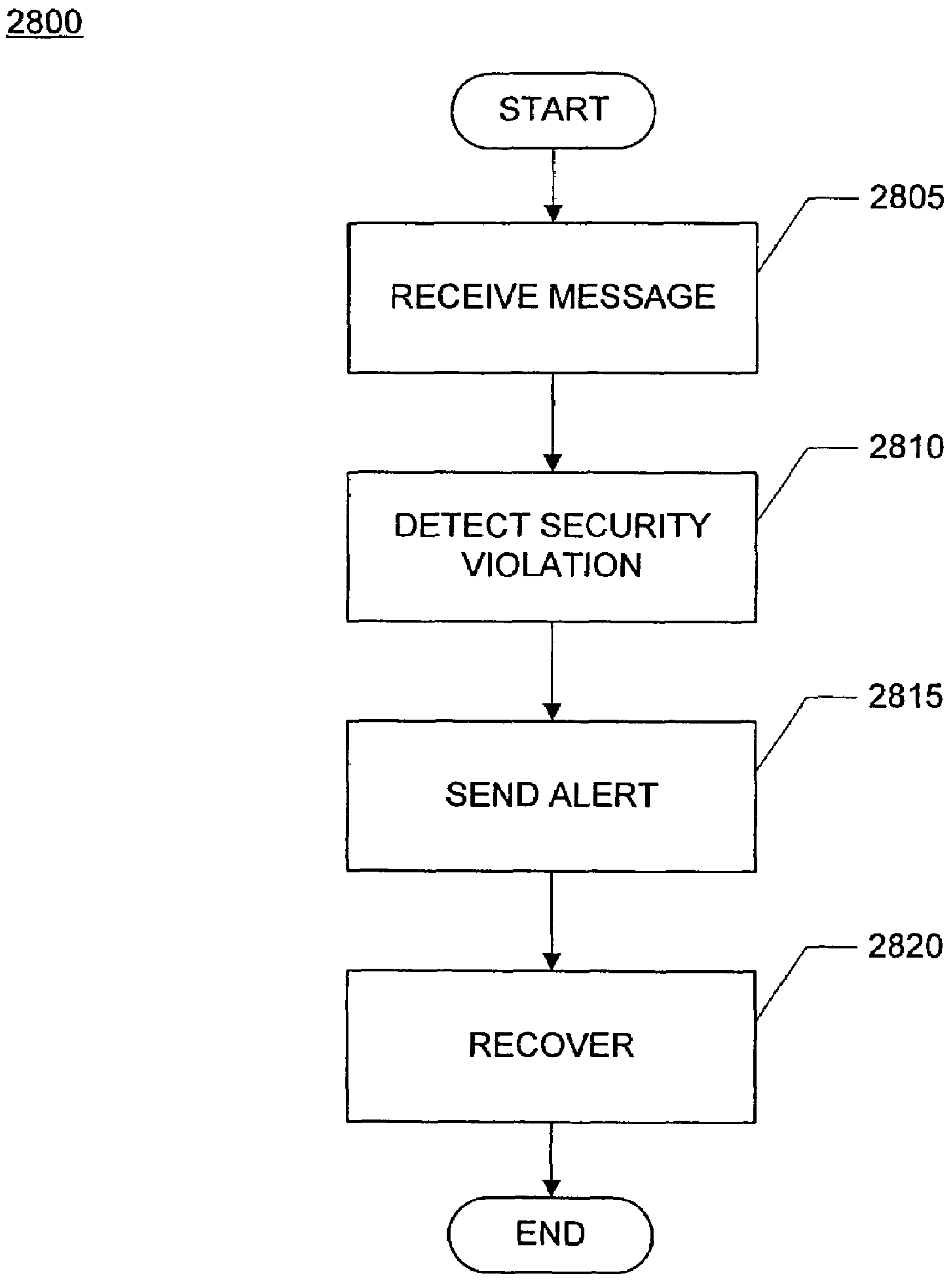

2800
START
2805
RECEIVE MESSAGE
2810
DETECT SECURITY VIOLATION
2815
SEND ALERT
2820
RECOVER
END

FIG. 29

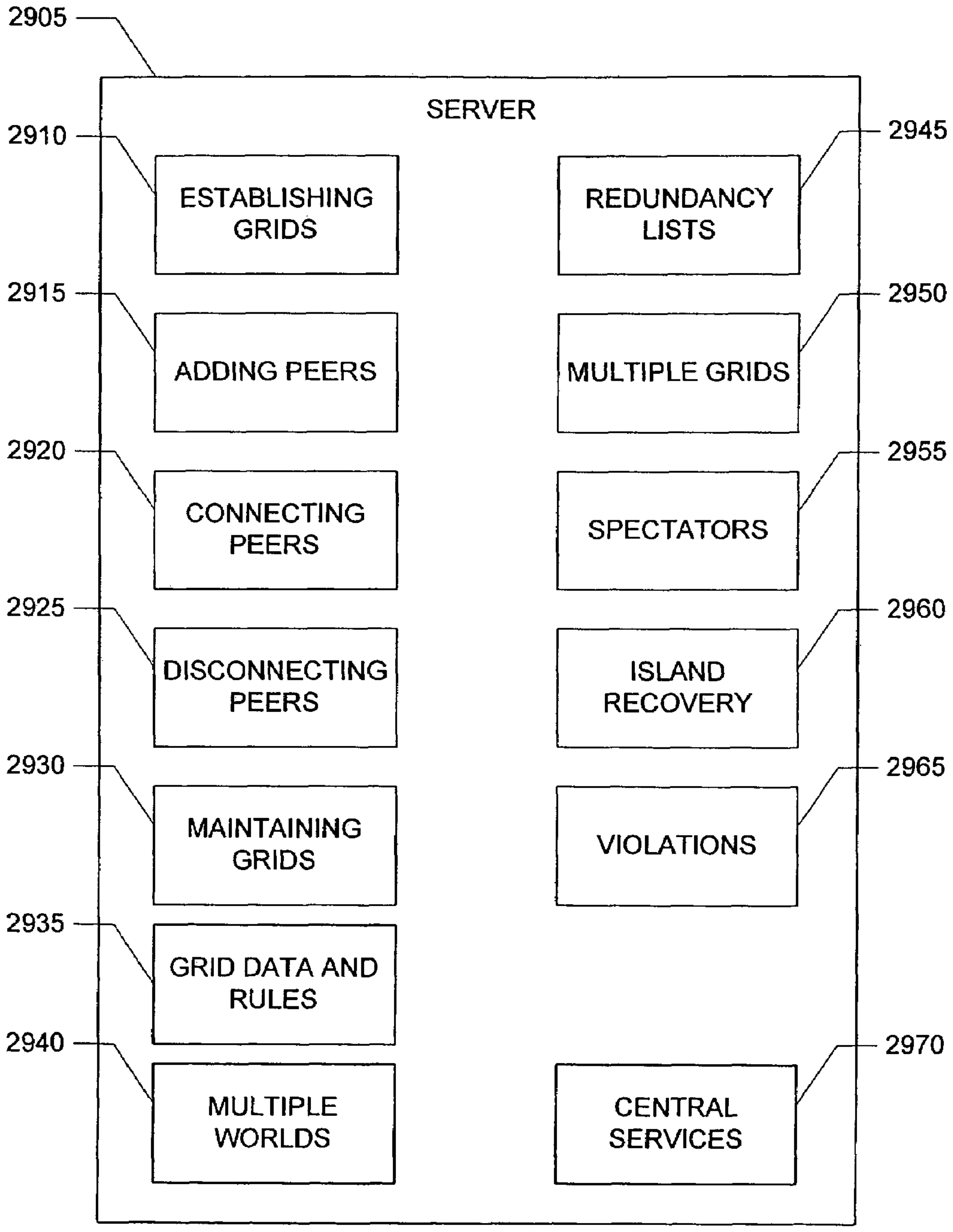

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

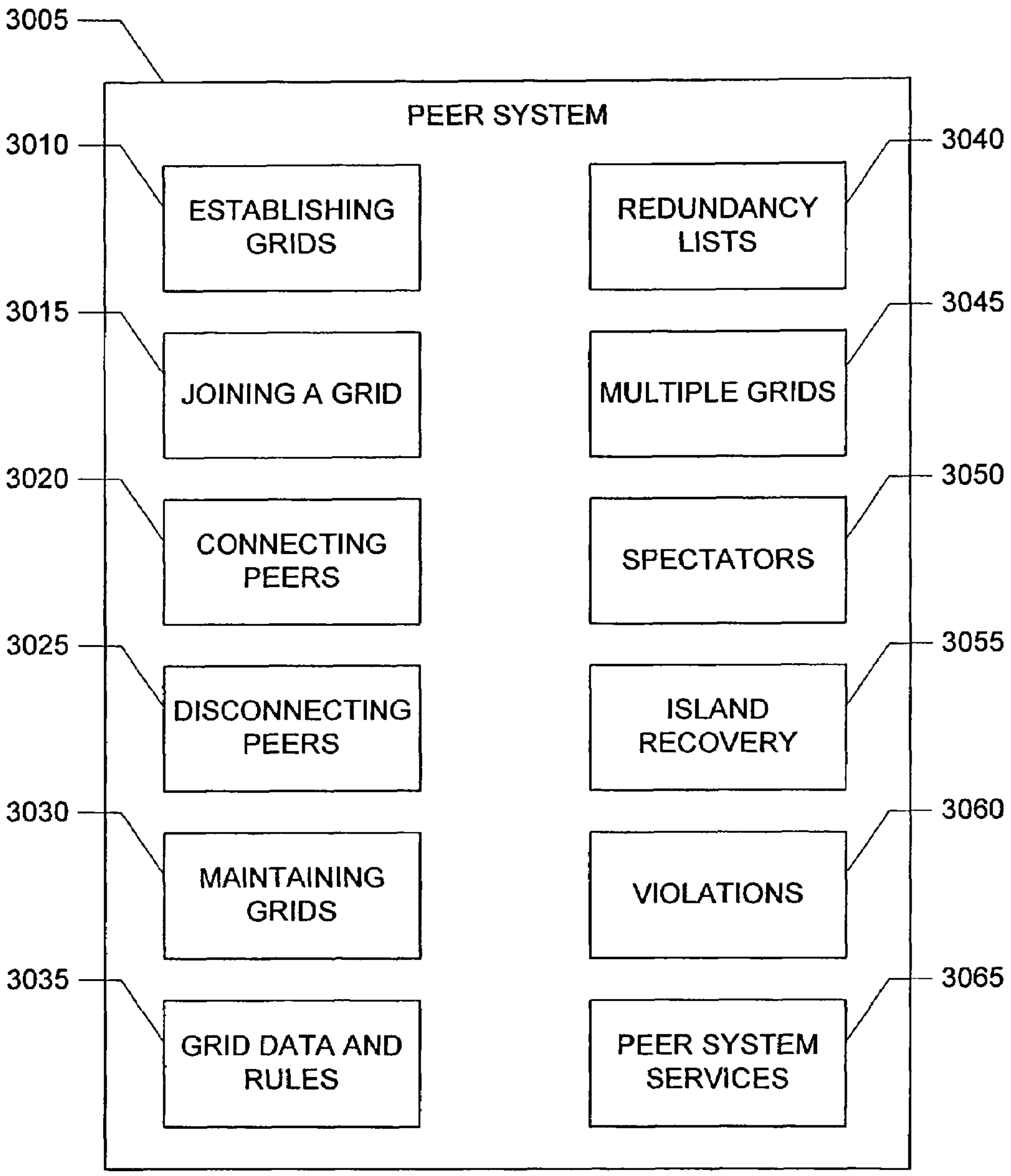

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

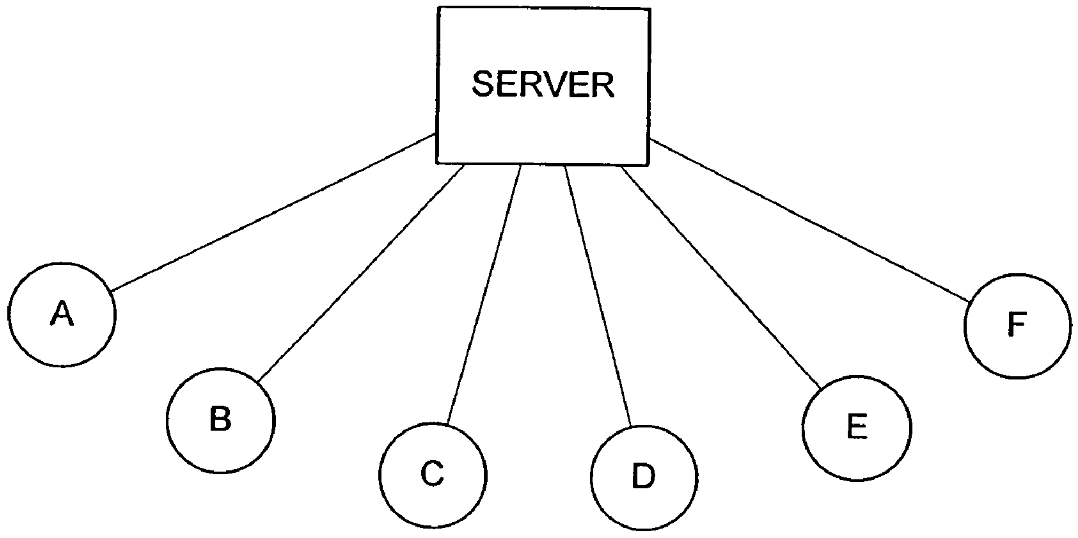
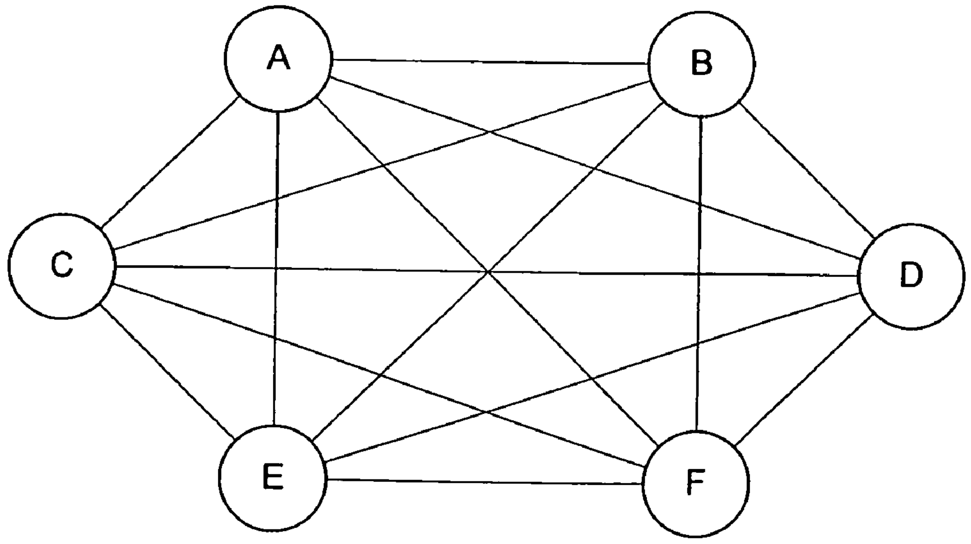


FIG. 31B



VIOLATIONS IN A PEER-TO-PEER RELAY NETWORK

This is a continuation of application Ser. No. 10/700,797, filed Nov. 3, 2003, now U.S. Pat. No. 7,392,422 with a claim of priority to Provisional Application 60/513,098, filed on Oct. 20, 2003, the entirety thereof being incorporated herein by reference.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N-1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of detecting and recovering from violations in a peer-to-peer relay network includes: receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network detecting a violation in said received message; and sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

In one implementation, a peer system in a peer-to-peer relay network includes: means for receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network; means for detecting a violation in said received message; and means for sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems $\mathbf{105}_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent applications Ser. Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed Jul. 31, 2002, and 10/------("Multi-User Application Programming Interface"), filed ------, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system $\mathbf{105}_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or

additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will

only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins, peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island

includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the

grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method of detecting and recovering from violations in a peer-to-peer relay network, comprising:

receiving a first message having first content data at a receiving peer system from a first sending peer system connected to the receiving peer system in the peer-to-peer relay network;

detecting a manipulation of data in said received first message, said manipulation of data changing the outcome of processing by the receiving peer system;

receiving a second message having second content data at the receiving peer system from at least one second sending peer system, wherein the second content data are expected to be substantially the same as the first content data;

wherein detecting the manipulation includes:

comparing by the receiving peer system the received first content data to the received second content data; and

determining whether the first message from the first sending peer system is different from at least one of the second messages based on the comparison;

when the first message from the first sending peer system is different, sending by the receiving peer system a manipulated data alert message to other peer systems connected to said receiving peer system in said peer-to-peer relay network, the manipulated data alert message identifying the first sending peer as responsible for the manipulation of data and not sending the message

received from the first peer system to other peer systems connected to said receiving peer system in said peer-to-peer relay network; and

when the first message received from the first sending peer system is not different, sending by the receiving peer system the first message to other peer systems connected to said receiving peer system in said peer-to-peer relay network,

wherein the receiving peer does not originate the first or second content data sent from the respective first or second sending peer system.

2. The method of claim **1**, further comprising:

causing the first sending peer system to disconnect from the peer-to-peer relay network.

3. The method of claim **1**, further comprising:

sending the manipulated data alert message to a server connected to the peer-to-peer relay network.

4. The method of claim **3**, wherein the server causes the first sending peer to disconnect from the peer-to-peer relay network.

5. The method of claim **1**, further comprising, sending a replacement message with second content data based on the second content data received from the at least one second sending peer.

6. The method of claim **1**, further comprising:

ignoring further messages sent by said first sending peer system.

7. The method of claim **1**, further comprising:

estimating by the receiving peer a correct content data for the first received message; and

relaying the correct data to other peers on the peer-to-peer network.

8. The method of claim **1**, further comprising:

the data relayed by peer systems is update data for a network environment.

9. The method of claim **1**, wherein the data relayed by peer systems is update data for an online game.

10. The method of claim **1**, wherein at least one peer system is a network-enabled game console.

11. The method of claim **1**, wherein at least two peer systems are connected through the Internet.

12. The method of claim **1**, wherein detecting said violation includes detecting invalid data in said first message.

13. The method of claim **1**, wherein detecting said violation includes detecting said first message was sent using improper sending procedures.

14. The method of claim **13**, wherein said first message was sent as part of denial of service attack.

15. A method of detecting and recovering from a cheating violation in a peer-to-peer relay network, comprising:

receiving a message having content data at a receiving peer system from a sending peer system connected to the receiving peer system in the peer-to-peer relay network;

detecting a manipulation of data in said received message, said manipulation of data changing the outcome of processing by the receiving peer system,

wherein detecting said cheating violation includes:

generating predicted data;

comparing by the receiving peer system the message from the sending peer system with the predicted data; and

determining whether the message received from the sending peer system is different from the predicted data; and

when the message from the sending peer system is different based on the predicted data, sending by the receiving peer system a manipulated data alert message to other peer systems connected to the receiving peer system in the peer-to-peer relay network, the manipulated data alert message identifying the sending peer as responsible for the manipulation of data and not sending the message received from the sending peer system to other peer systems connected to said receiving peer system in said peer-to-peer relay network; and

when the message received from the sending peer system is not different, sending by the receiving peer system the message to other peer systems connected to said receiving peer system in said peer-to-peer relay network,

wherein the receiving peer does not originate the content data sent from the sending peer system.

16. The method of claim **15**, further comprising:

sending the predicted data to each other peer system connected to the peer system in the peer-to-peer relay network.

17. A receiving peer system in a peer-to-peer relay network, comprising:

means for receiving a first message having first content data at the receiving peer system from a first sending peer system connected to said peer system in a peer-to-peer relay network;

means for detecting a manipulation of data in said received first message, said manipulation of data changing the outcome of processing by the receiving peer system;

means for receiving a second message having second content data at the receiving peer system from at least one second sending peer system, wherein the second content data are expected to be substantially the same as the first content data;

wherein detecting said manipulation includes:

comparing by the receiving peer system the received first content data to the received second content data;

determining whether the first message from the first sending peer system is different from at least one of the second messages based on the comparison; and

when the message from the first sending peer system is different, sending by the receiving peer system a manipulated data alert message to other peer systems connected to said receiving peer system in said peer-to-peer relay network, the manipulated data alert message identifying the first sending peer as responsible for the manipulation of data and not sending the message received from the first peer system to other peer systems connected to said receiving peer system in said peer-to-peer relay network; and

when the first message received from the first sending peer system is not different, sending by the receiving peer system the first message to other peer systems connected to said receiving peer system in said peer-to-peer relay network.

18. The peer system of claim **17**, further comprising:

means for sending said data manipulation alert message to a server connected to said peer system.

19. The method of claim **18**, wherein the server causes the first sending peer to disconnect from the peer-to-peer relay network.

20. A computer-readable medium storing a computer-readable program that when executed on a processor causes the processor to execute a method in a peer system of a peer-to-peer relay network, the method comprising the steps of:

receiving a first message having first content data at a receiving peer system from a first sending peer system connected to said peer system in a peer-to-peer relay network

detecting a manipulation of data in said received first message, said manipulation of data changing the outcome of processing by the receiving peer system;

receiving a second message having second content data at the receiving peer system from at least one second sending peer system, wherein the second content data are expected to be substantially the same as the first content data;

wherein detecting the manipulation includes:

comparing by the receiving peer system the received first content to the received second content data; and

second content data; and

determining whether the first message from the first sending peer system is different from at least one of the second messages based on the comparison; and

when the message from the first sending peer system is different, sending by the receiving peer system a manipulated data alert message to other peer systems connected to said receiving peer system in said peer-to-peer relay network, the manipulated data alert message identifying the first sending peer as responsible for the manipulation of data and not sending the message received from the first peer system to other peer systems connected to said receiving peer system in said peer-to-peer relay network; and

when the first message received from the first sending peer system is not different, sending by the receiving peer system the first message to other peer systems connected to said receiving peer system in said peer-to-peer relay network,

wherein the receiving peer does not originate the first or second content data sent from the respective first or second sending peer system.

21. The computer-readable medium of claim **20**, further comprising sending said data manipulation alert message to a server connected to said peer system.

22. The method of claim **21**, wherein the server causes the first sending peer to disconnect from the peer-to-peer relay network.

* * * * *

EXHIBIT 86

US007392422B2

(12) **United States Patent**
Van Datta et al.

(10) **Patent No.: US 7,392,422 B2**
(45) **Date of Patent: Jun. 24, 2008**

(54) VIOLATIONS IN A PEER-TO-PEER RELAY NETWORK

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.,**, Foster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 438 days.

(21) Appl. No.: **10/700,797**

(22) Filed: **Nov. 3, 2003**

(65) **Prior Publication Data**

US 2005/0097386 A1 May 5, 2005

Related U.S. Application Data

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/4**

(58) **Field of Classification Search** 714/819, 714/920, 4, 716, 820, 49; 713/187, 176, 713/179, 181; 340/5.8, 5.24; 726/10, 22, 726/23, 24, 110; 702/185; 700/30
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,325 A * 8/1996 Denny et al. 709/236
5,638,446 A * 6/1997 Rubin 705/51
5,768,382 A * 6/1998 Schneier et al. 380/251
5,835,726 A * 11/1998 Shwed et al. 709/229

(Continued)

FOREIGN PATENT DOCUMENTS

EP 0 913 965 5/1999

(Continued)

OTHER PUBLICATIONS

Baughman et al., Cheat-proof playout for centralized and distributed online games INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Publication Date: Apr. 22-26, 2001, On pp. 104-113 vol. 1.*

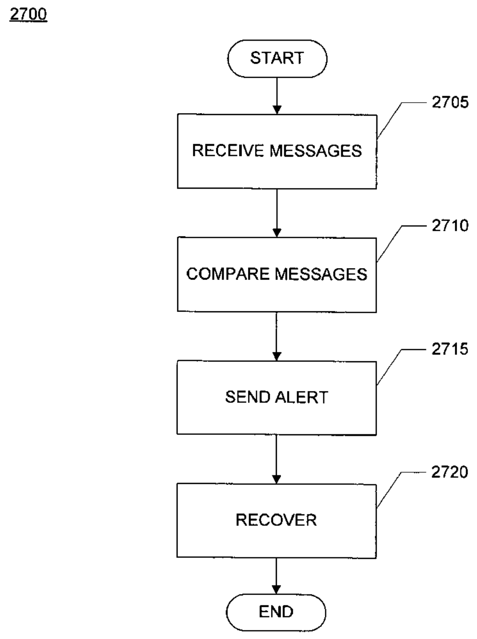
(Continued)

Primary Examiner—Robert Beausoliel
Assistant Examiner—Elmira Mehrmanesh
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) ABSTRACT

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of detecting and recovering from violations in a peer-to-peer relay network includes: receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network detecting a violation in said received message; and sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

3 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

6,119,229	A *	9/2000	Martinez et al.	726/28
6,327,630	B1 *	12/2001	Carroll et al.	719/314
6,405,104	B1 *	6/2002	Dougherty	700/292
6,779,017	B1 *	8/2004	Lamberton et al.	709/203
6,829,634	B1 *	12/2004	Holt et al.	709/204
6,899,628	B2 *	5/2005	Leen et al.	463/42
6,978,294	B1 *	12/2005	Adams et al.	709/217
7,043,641	B1 *	5/2006	Martinek et al.	713/187
7,168,089	B2 *	1/2007	Nguyen et al.	726/4
7,169,050	B1 *	1/2007	Tyler	463/42
7,203,841	B2 *	4/2007	Jackson et al.	713/187
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0107786	A1 *	8/2002	Lehmann-Haupt et al.	705/37
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1	12/2002	Traversat et al.	
2003/0028585	A1 *	2/2003	Yeager et al.	709/201
2003/0055892	A1	3/2003	Huitema et al.	
2003/0126245	A1 *	7/2003	Feltin et al.	709/223
2003/0229779	A1 *	12/2003	Morais et al.	713/153
2003/0229789	A1 *	12/2003	Morais et al.	713/171
2005/0020354	A1 *	1/2005	Nguyen et al.	463/25
2005/0086287	A1 *	4/2005	Datta	709/201
2005/0086288	A1 *	4/2005	Datta et al.	709/201
2005/0086329	A1 *	4/2005	Datta et al.	709/220
2005/0086350	A1 *	4/2005	Mai	709/230
2005/0086369	A1 *	4/2005	Mai et al.	709/239

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups" Proceedings of the 7th ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. CONF. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

* cited by examiner

FIG. 1

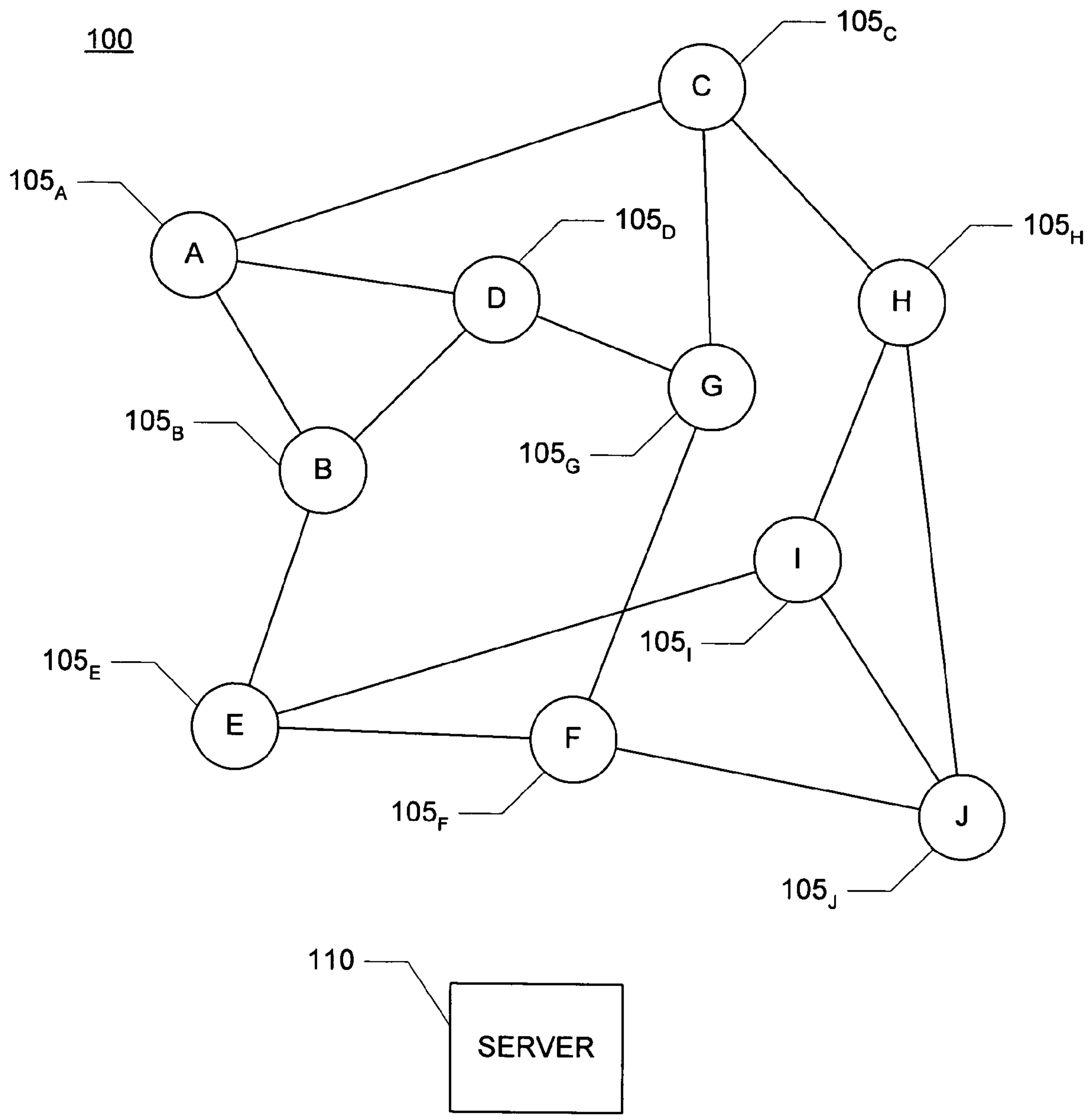

100
105C
C
105A
A
105D
D
105H
H
G
105B
B
105G
I
105I
105E
E
F
105F
J
105J
110
SERVER

FIG. 2

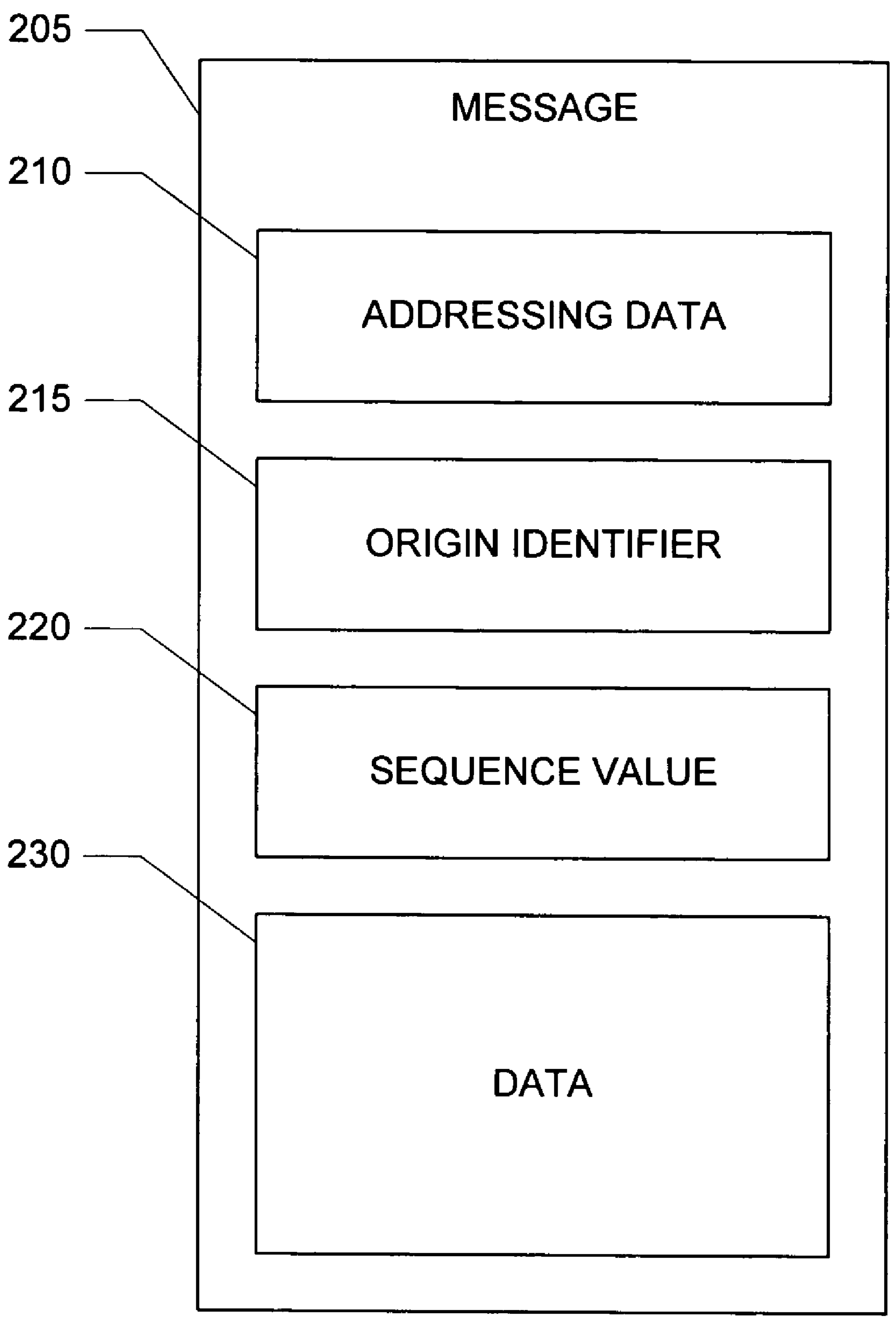

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

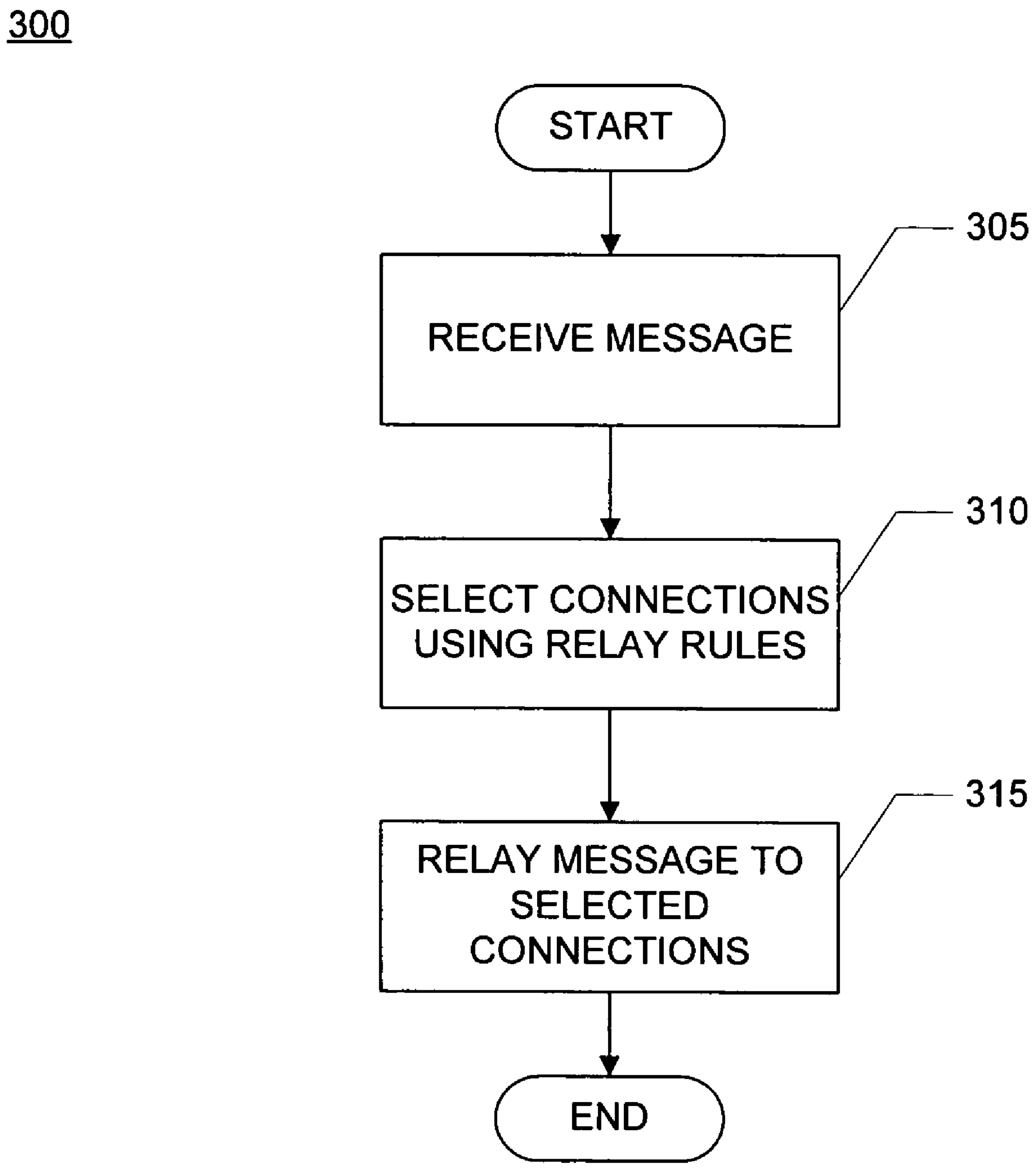

300
START
RECEIVE MESSAGE
305
SELECT CONNECTIONS USING RELAY RULES
310
RELAY MESSAGE TO SELECTED CONNECTIONS
315
END

FIG. 4

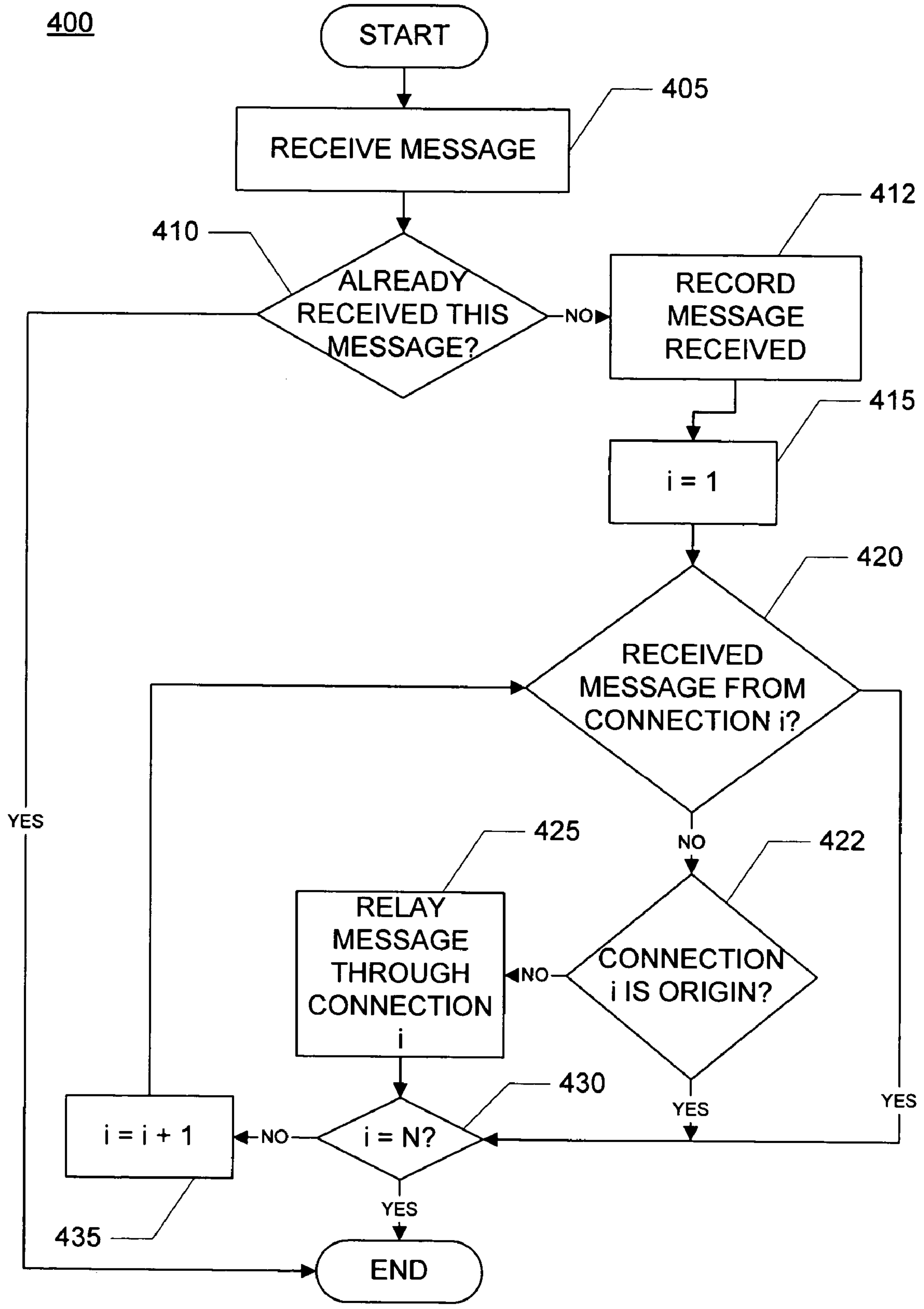

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
YES
RECORD MESSAGE RECEIVED
412
i = 1
415
RECEIVED MESSAGE FROM CONNECTION i?
420
NO
YES
CONNECTION i IS ORIGIN?
422
NO
YES
RELAY MESSAGE THROUGH CONNECTION i
425
i = N?
430
NO
YES
i = i + 1
435
END

FIG. 5

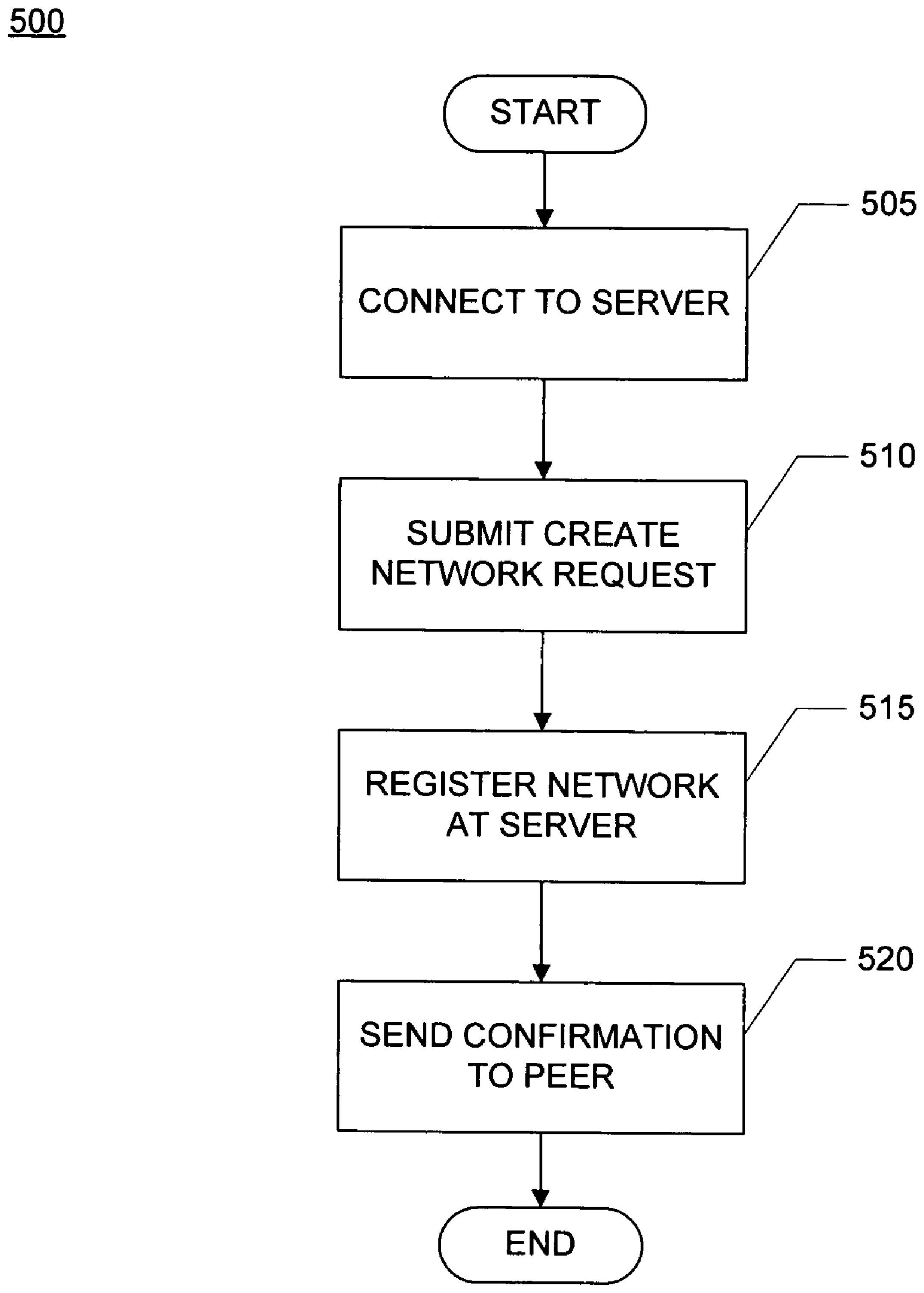

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

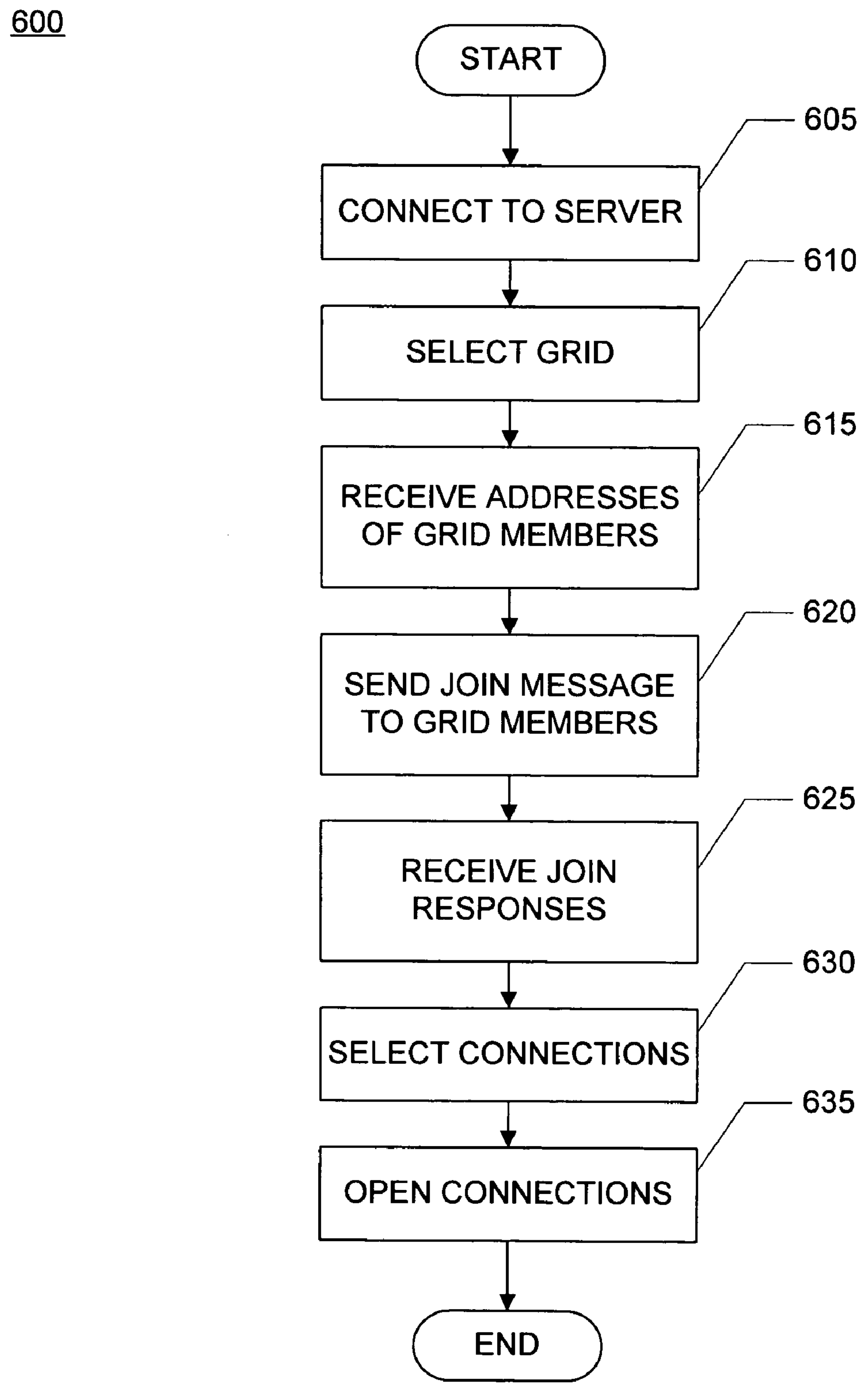

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

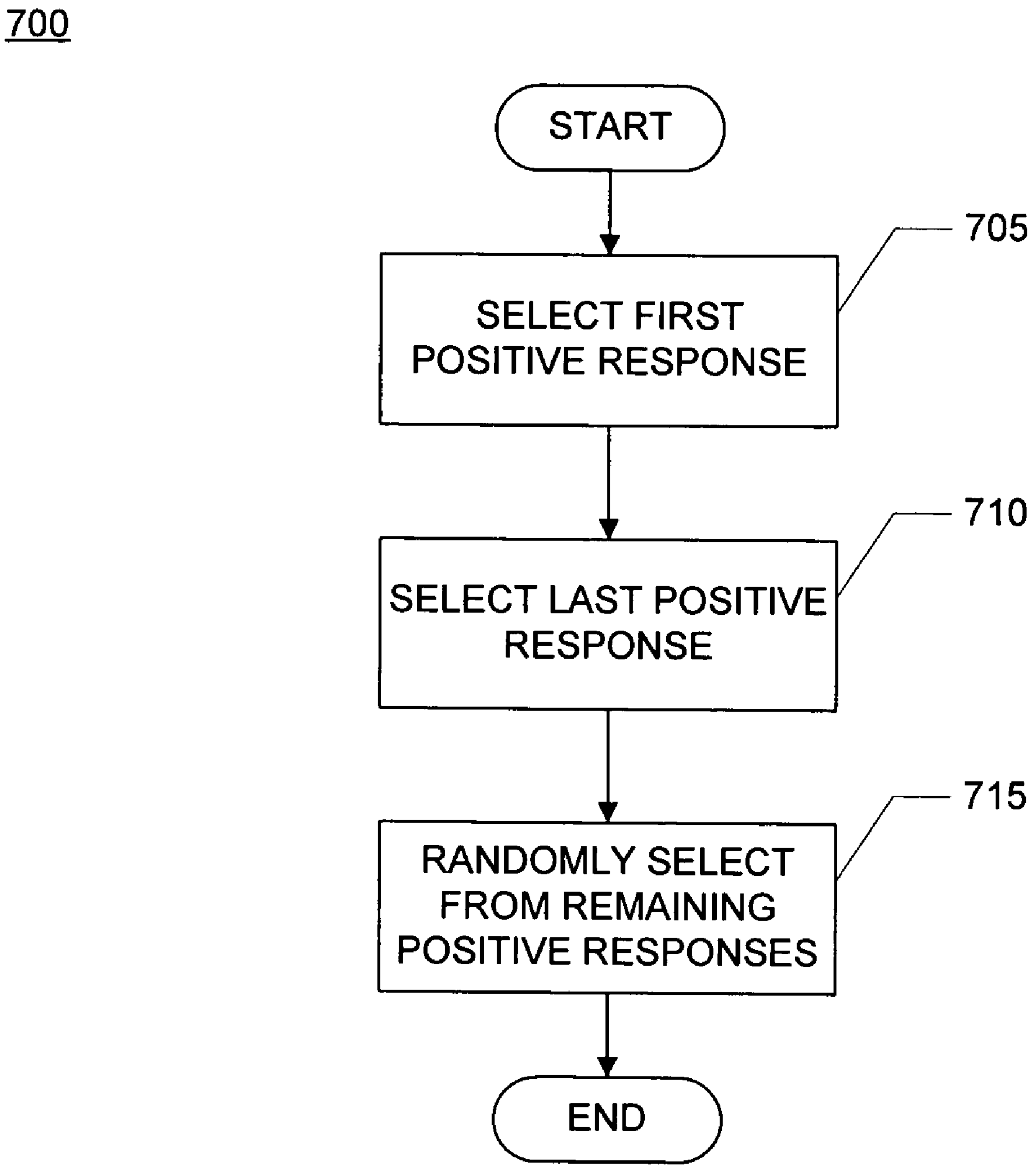

700
START
SELECT FIRST POSITIVE RESPONSE
705
SELECT LAST POSITIVE RESPONSE
710
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
715
END

FIG. 8

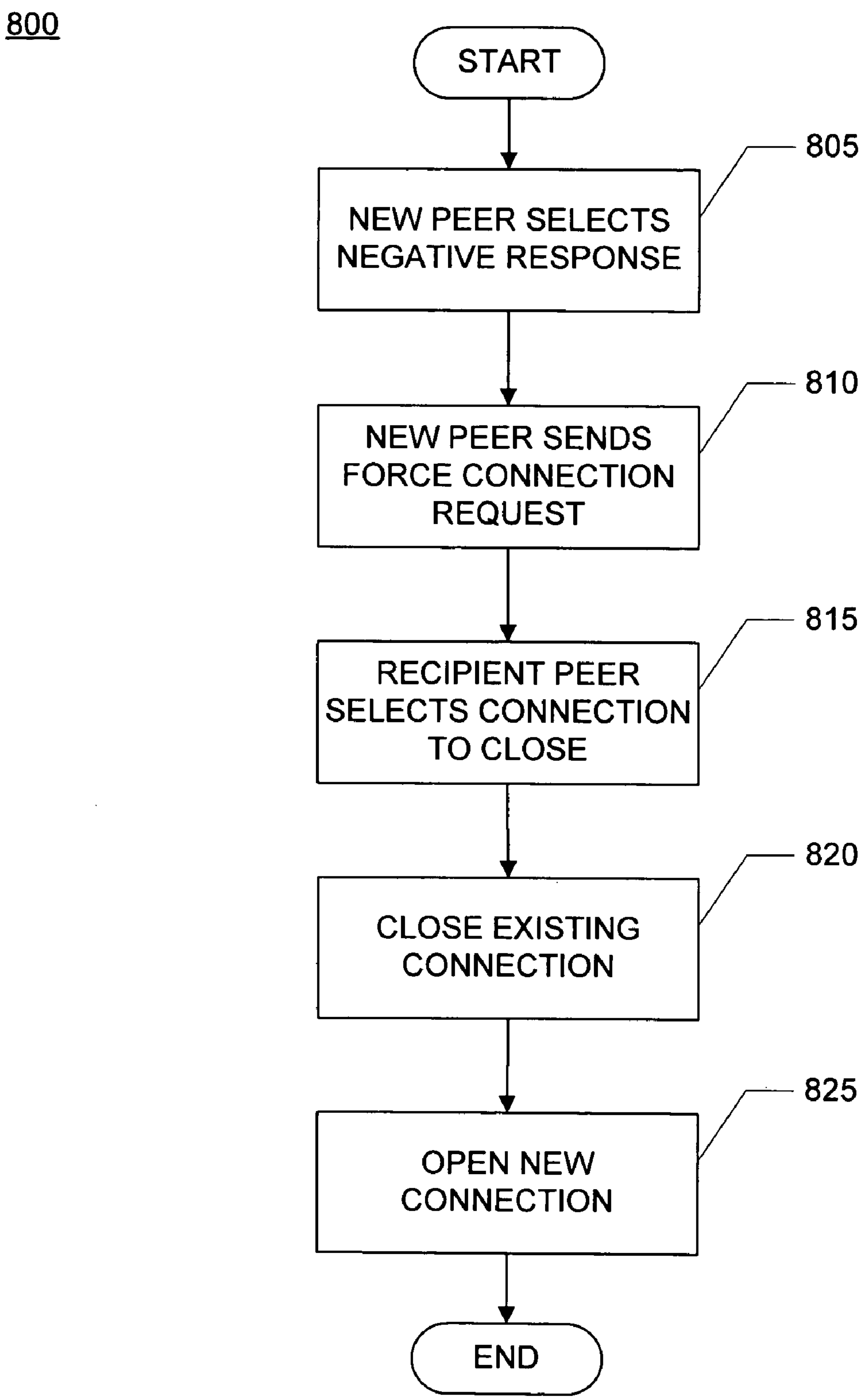

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

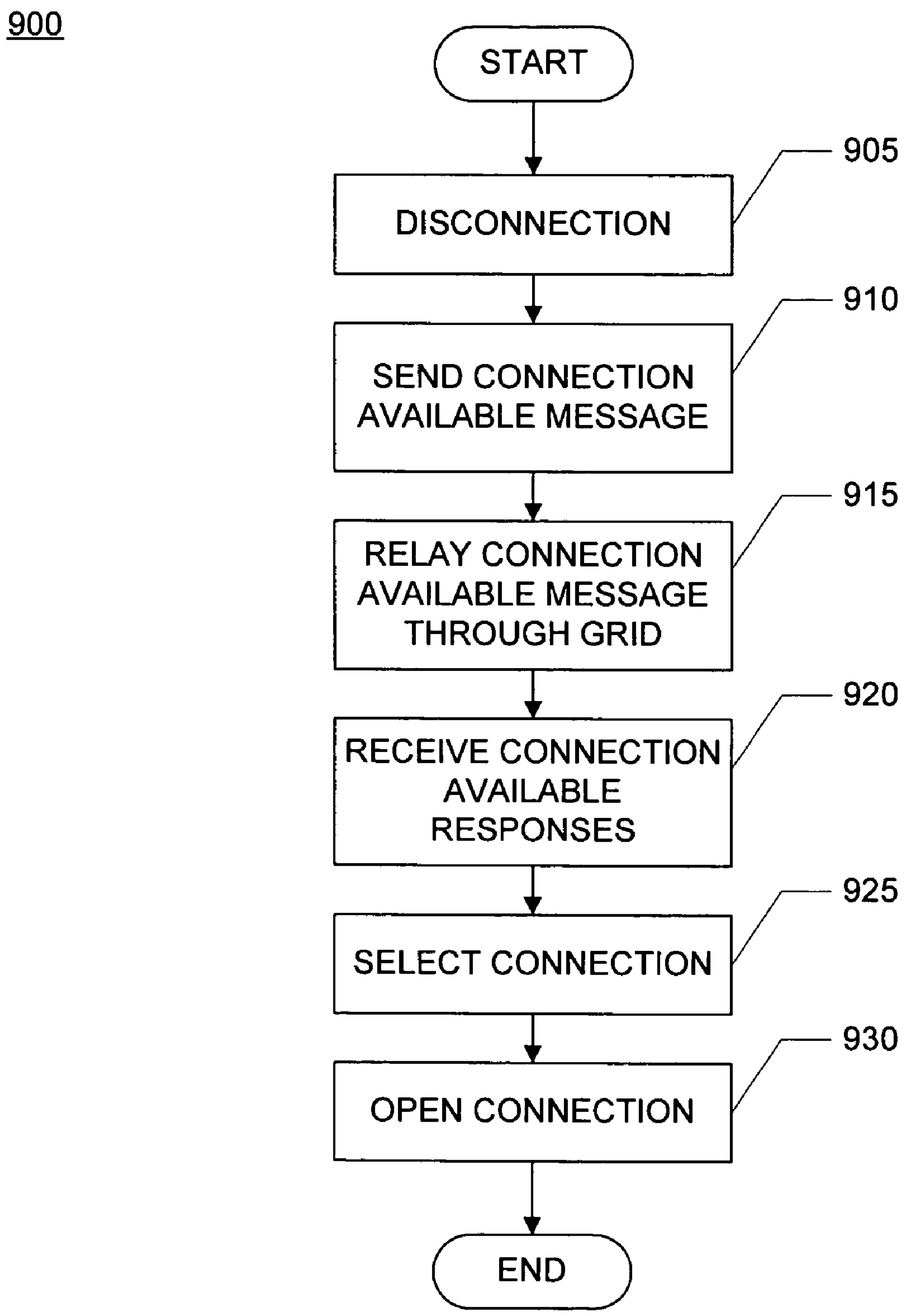

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

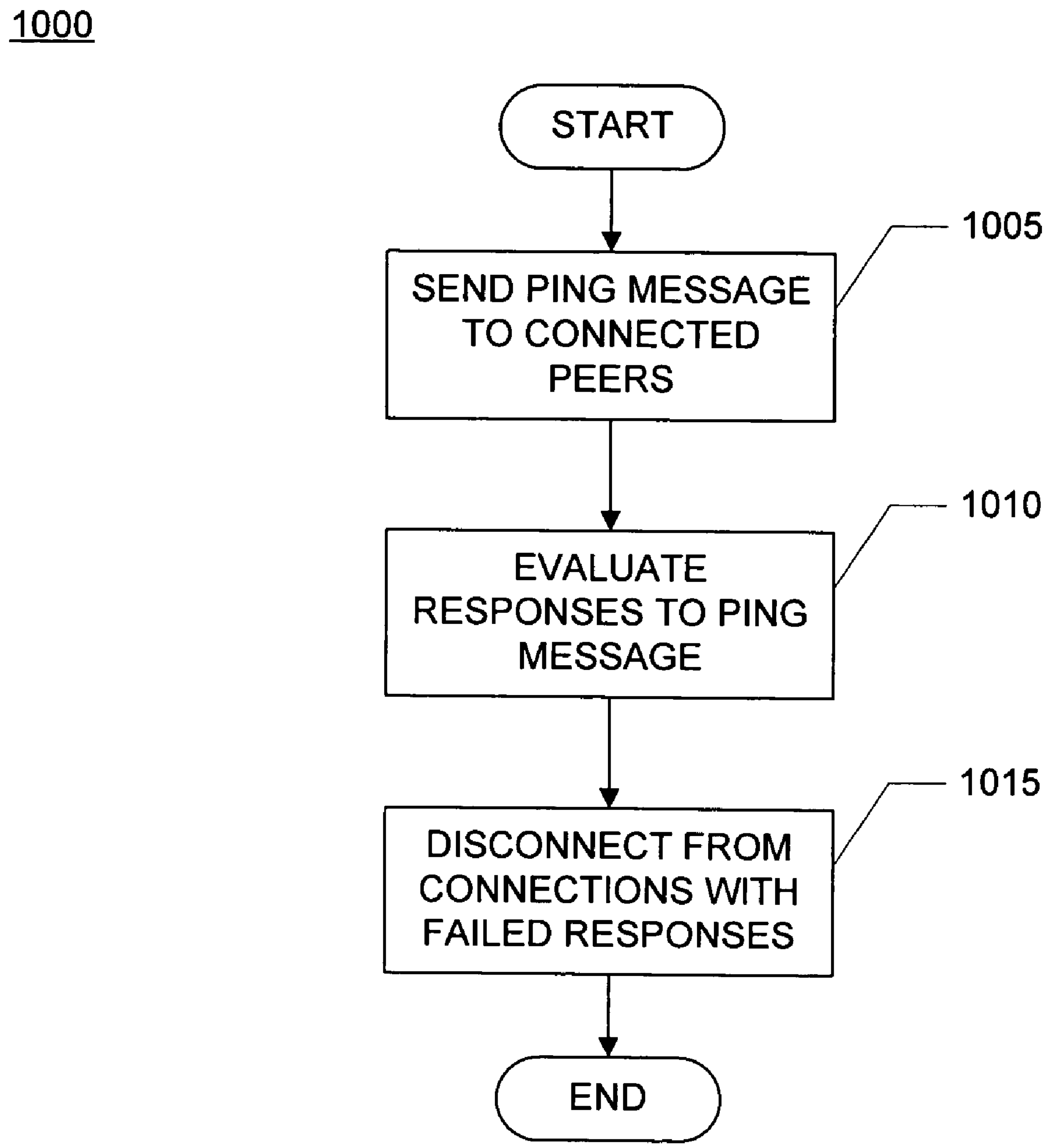

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

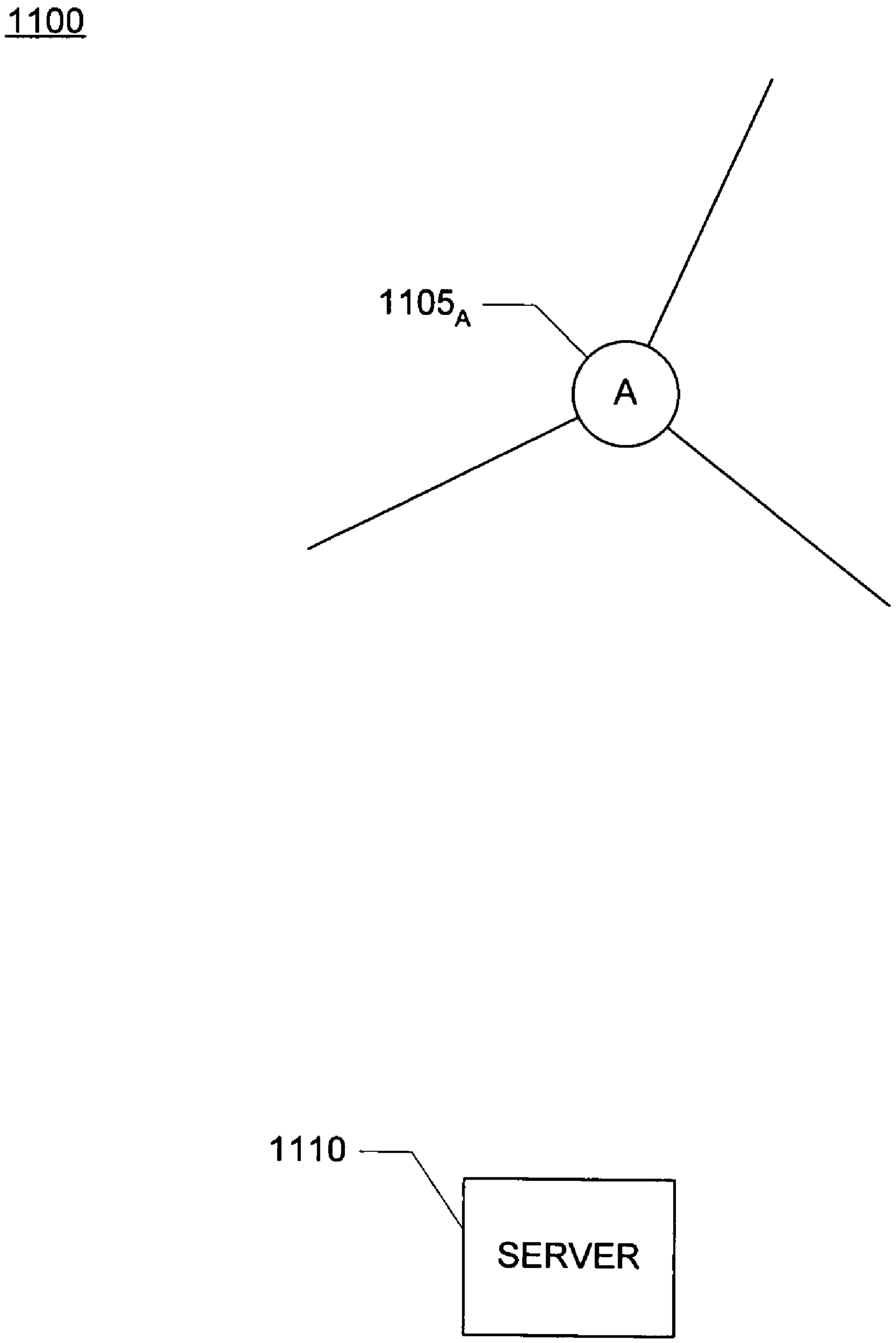


FIG. 12

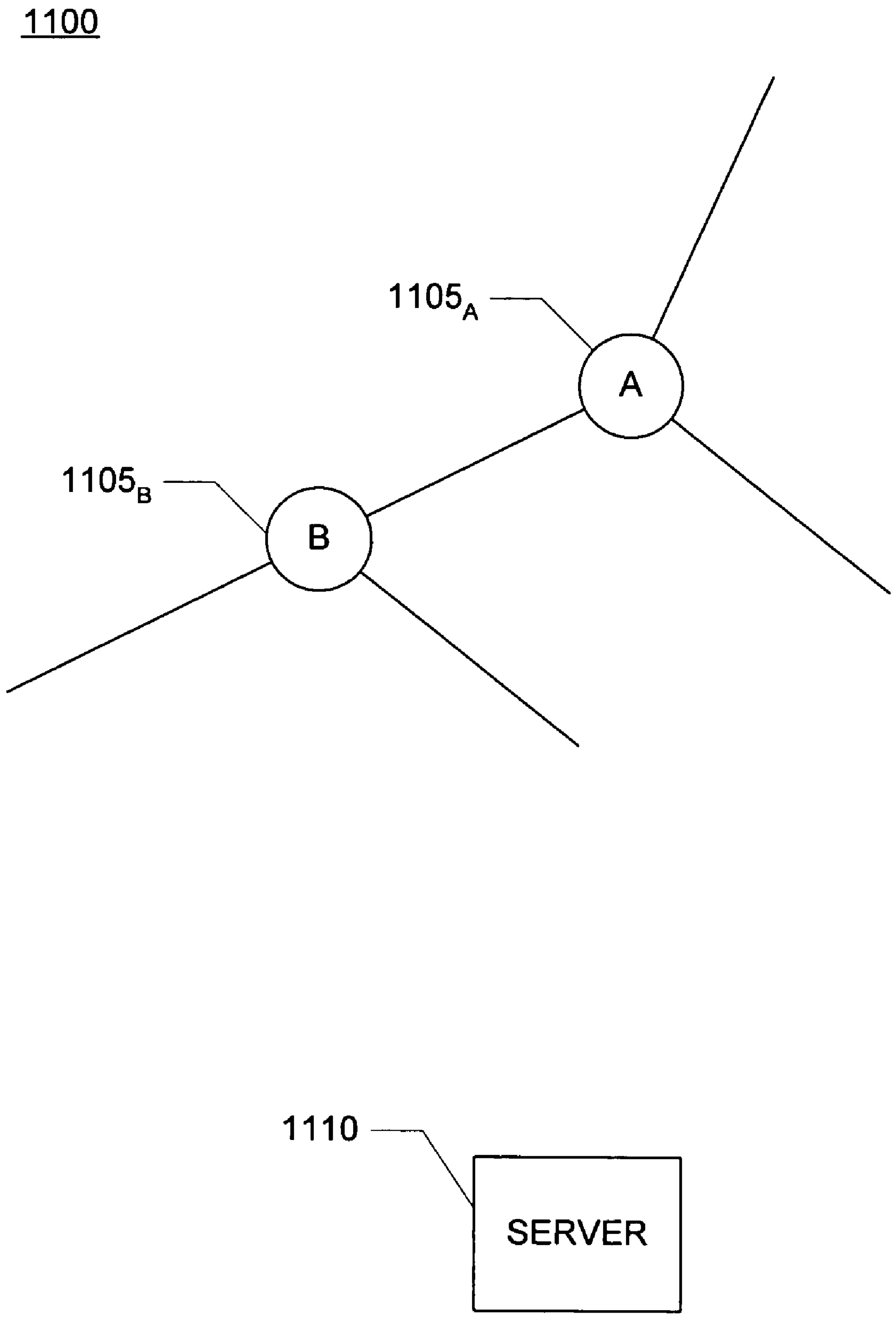


FIG. 13

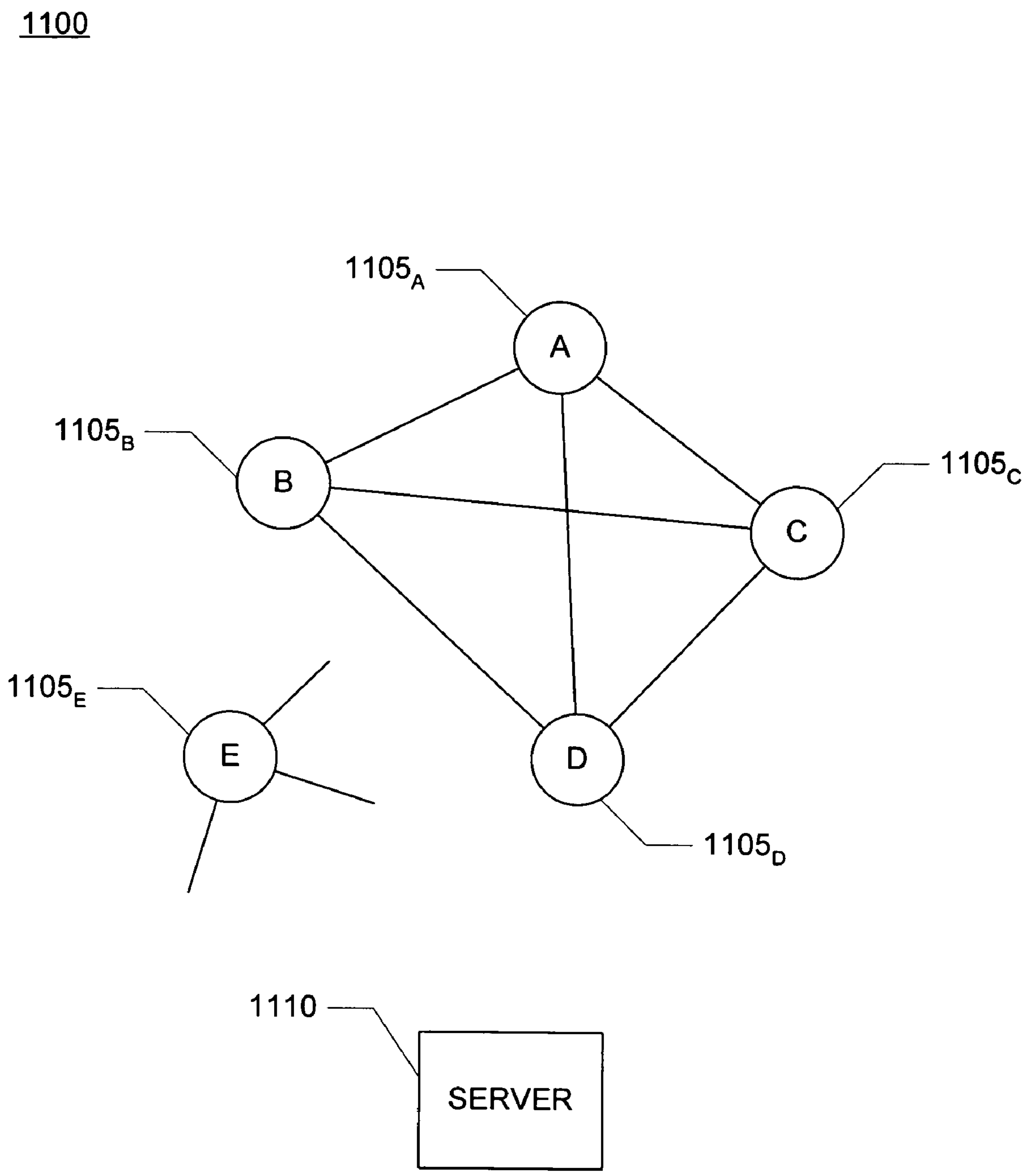


FIG. 14

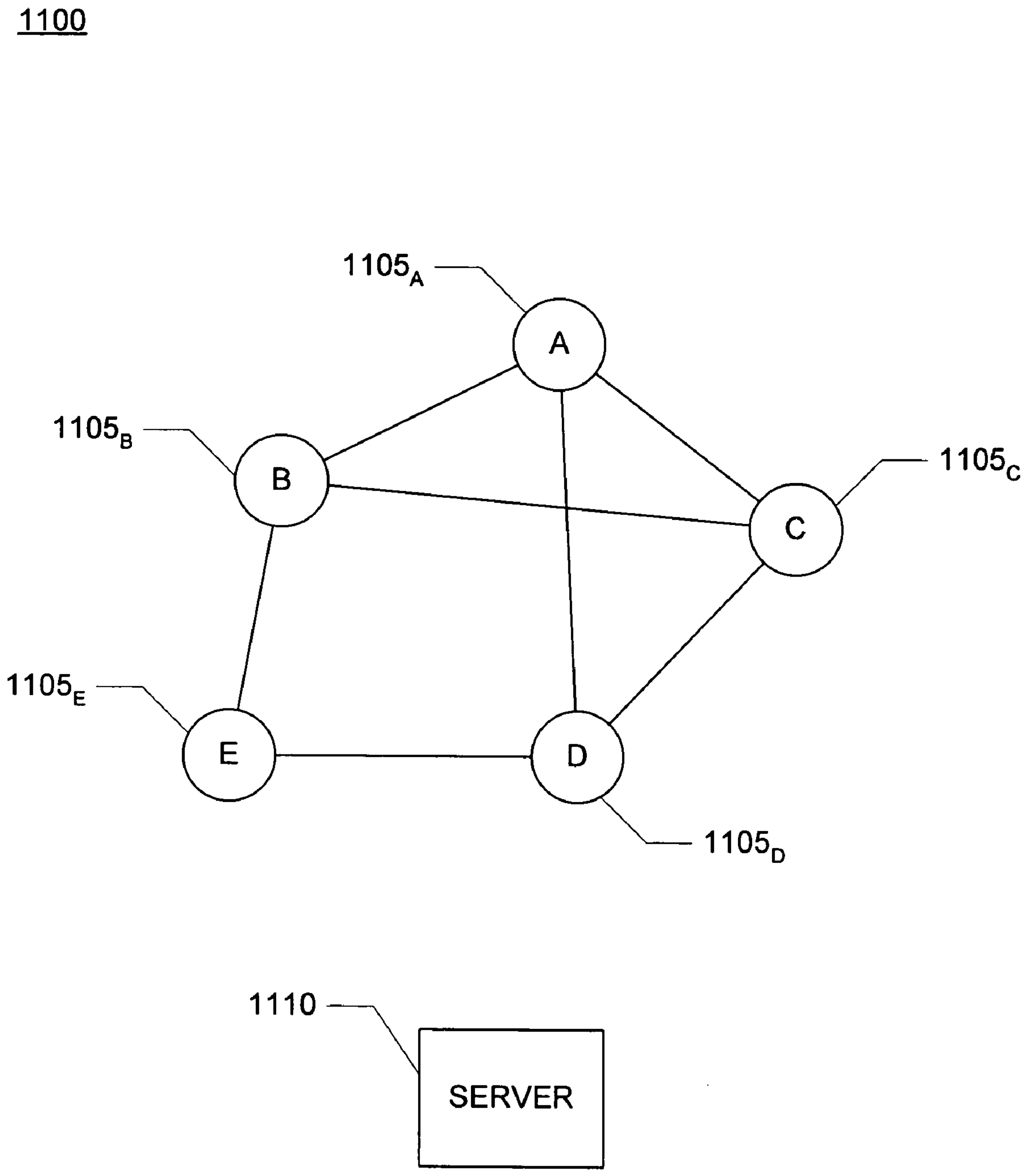


FIG. 15

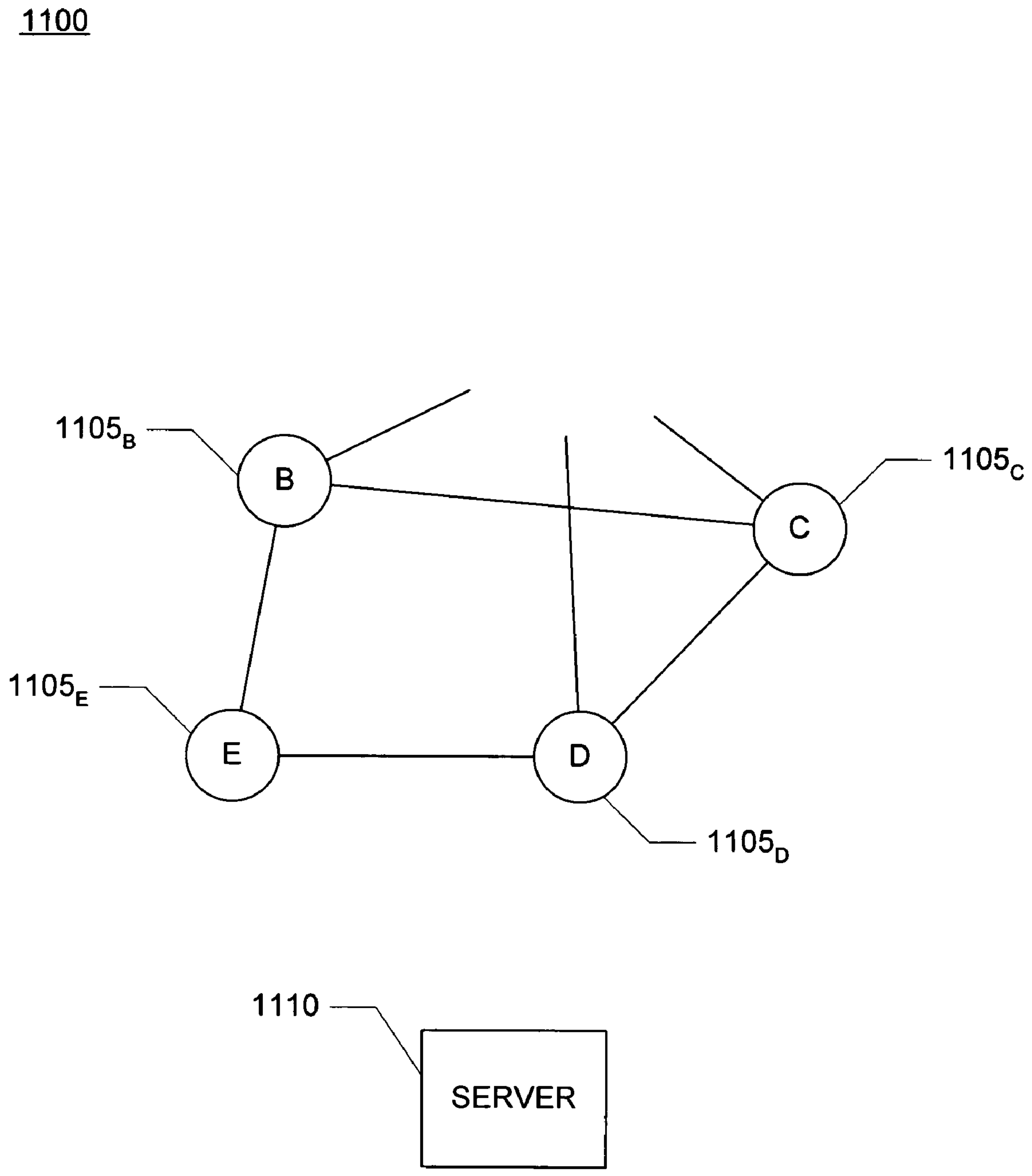


FIG. 16

1100

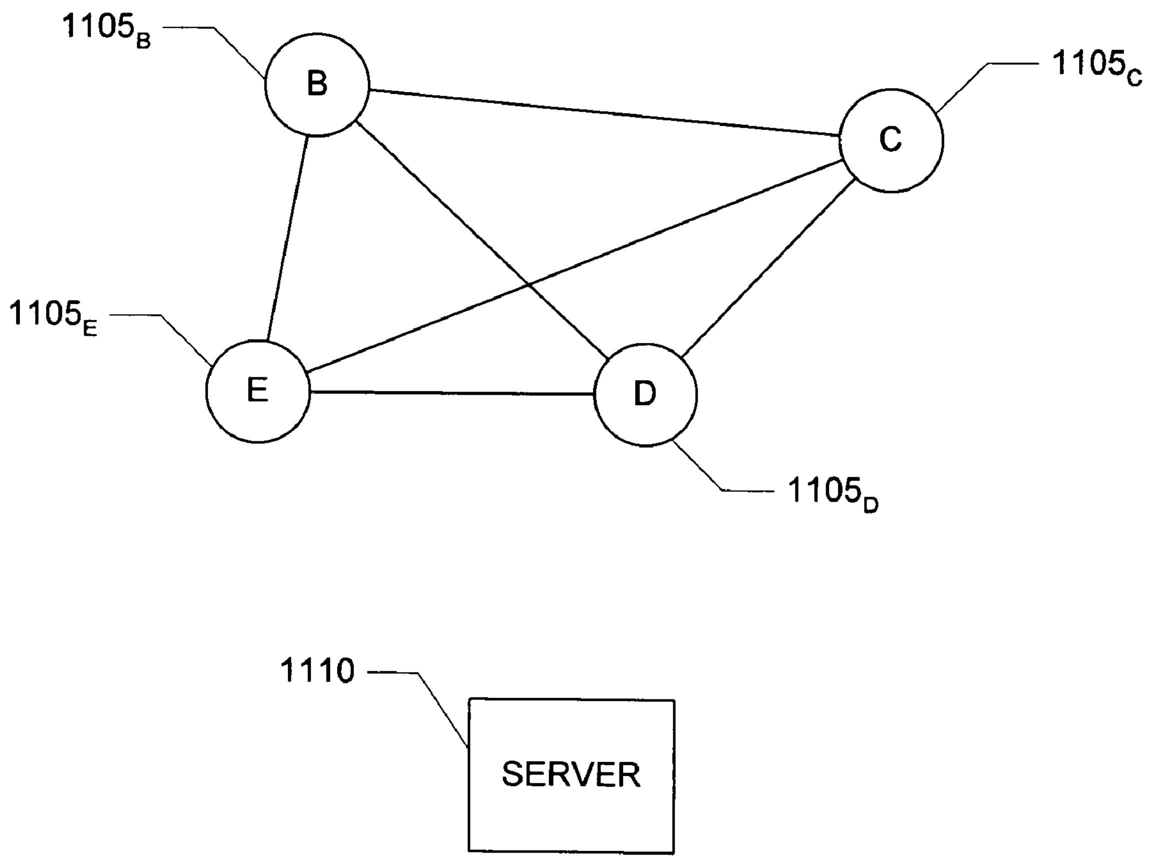


FIG. 17

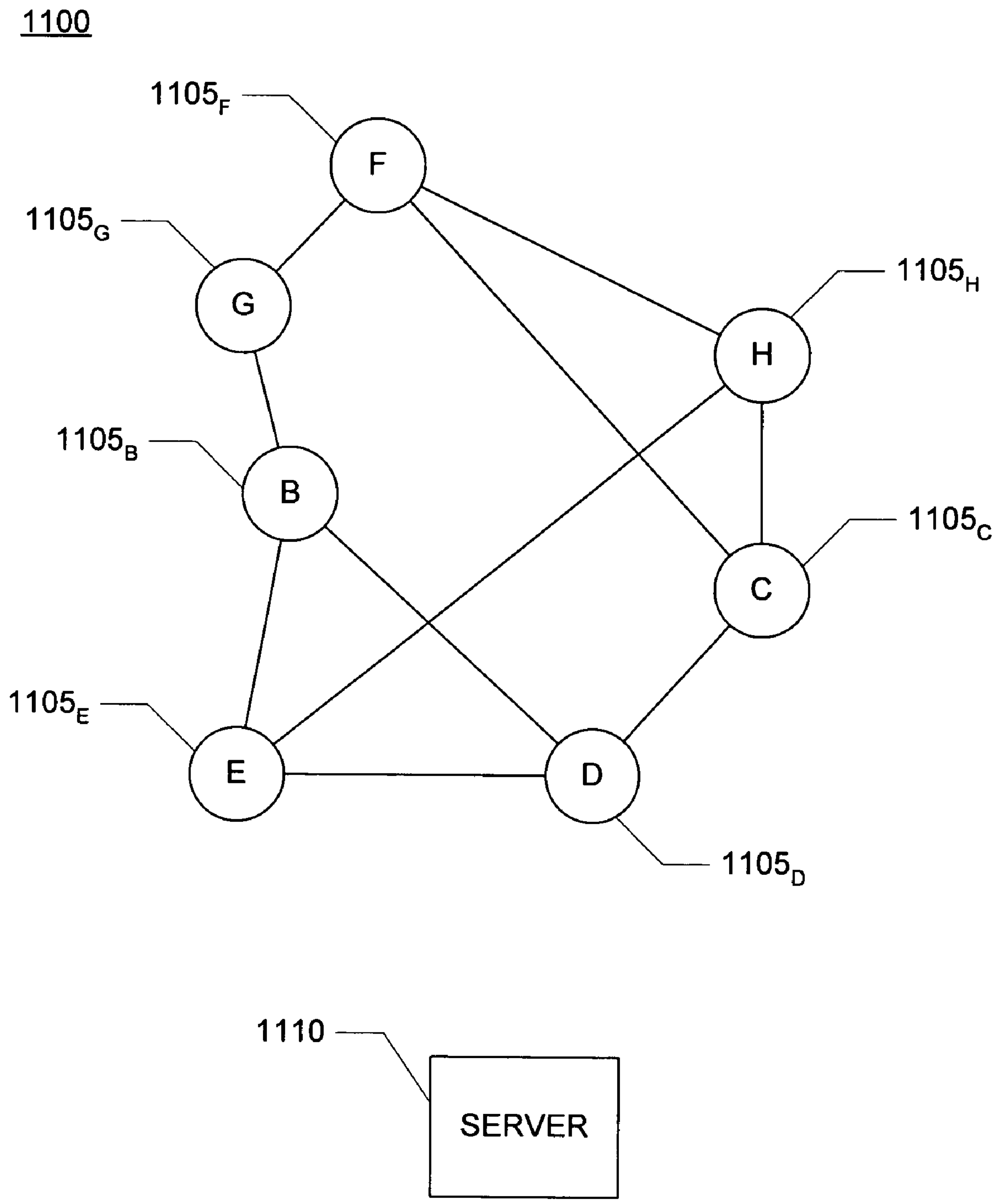

1100
1105$_F$
F
1105$_G$
G
1105$_H$
H
1105$_B$
B
1105$_C$
C
1105$_E$
E
D
1105$_D$
1110
SERVER

FIG. 18

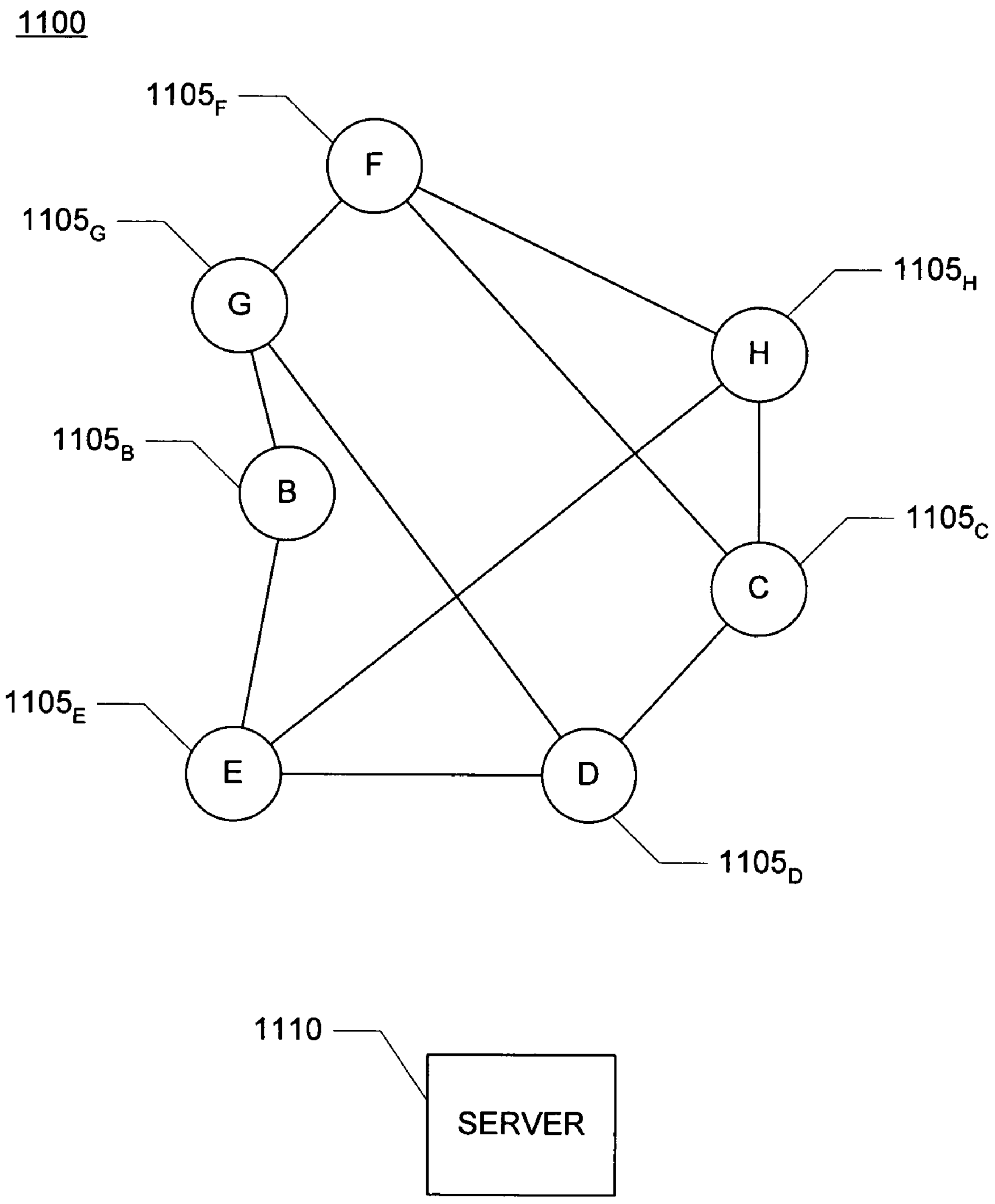

1100
1105$_F$
F
1105$_G$
G
1105$_H$
H
1105$_B$
B
1105$_C$
C
1105$_E$
E
D
1105$_D$
1110
SERVER

FIG. 19

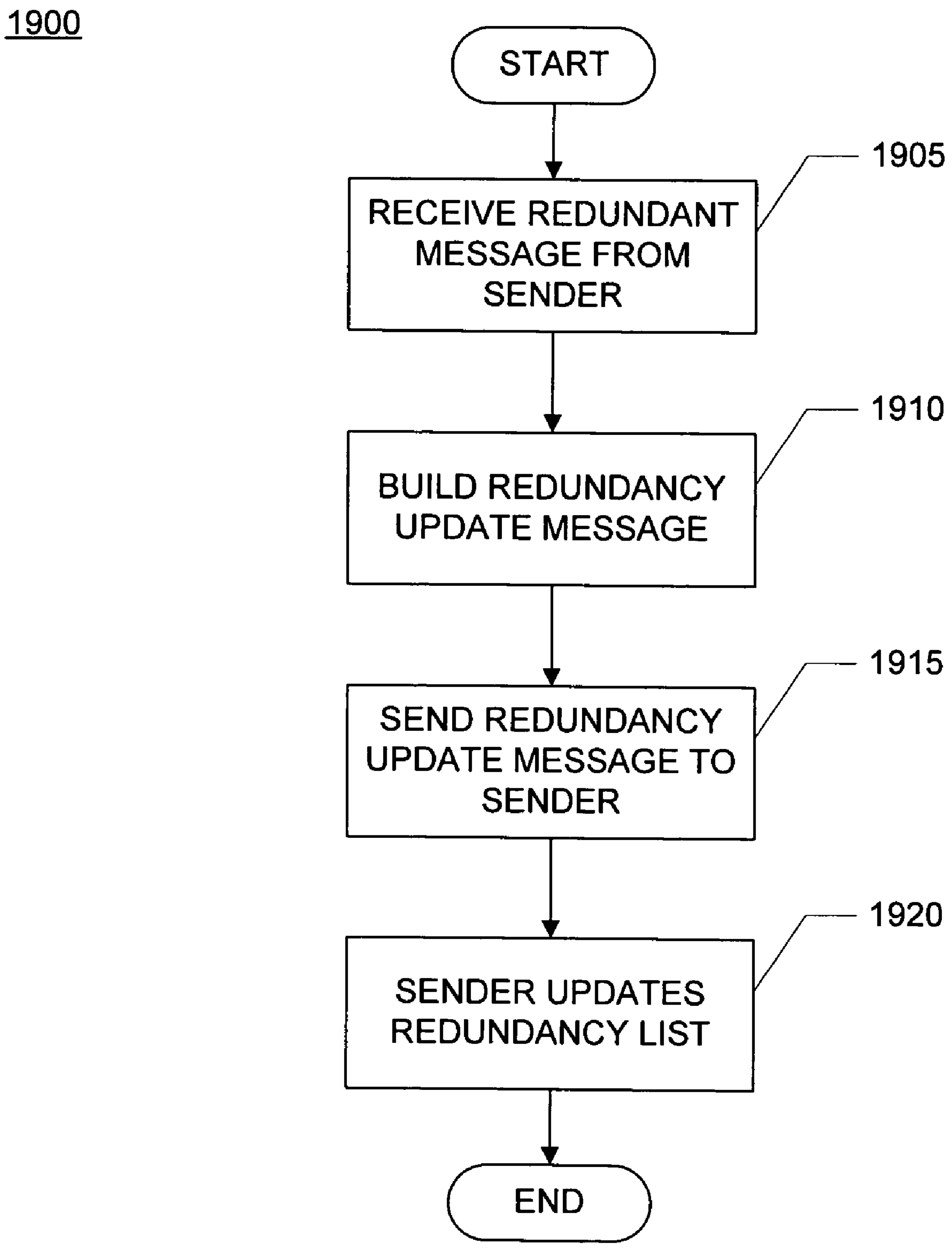

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

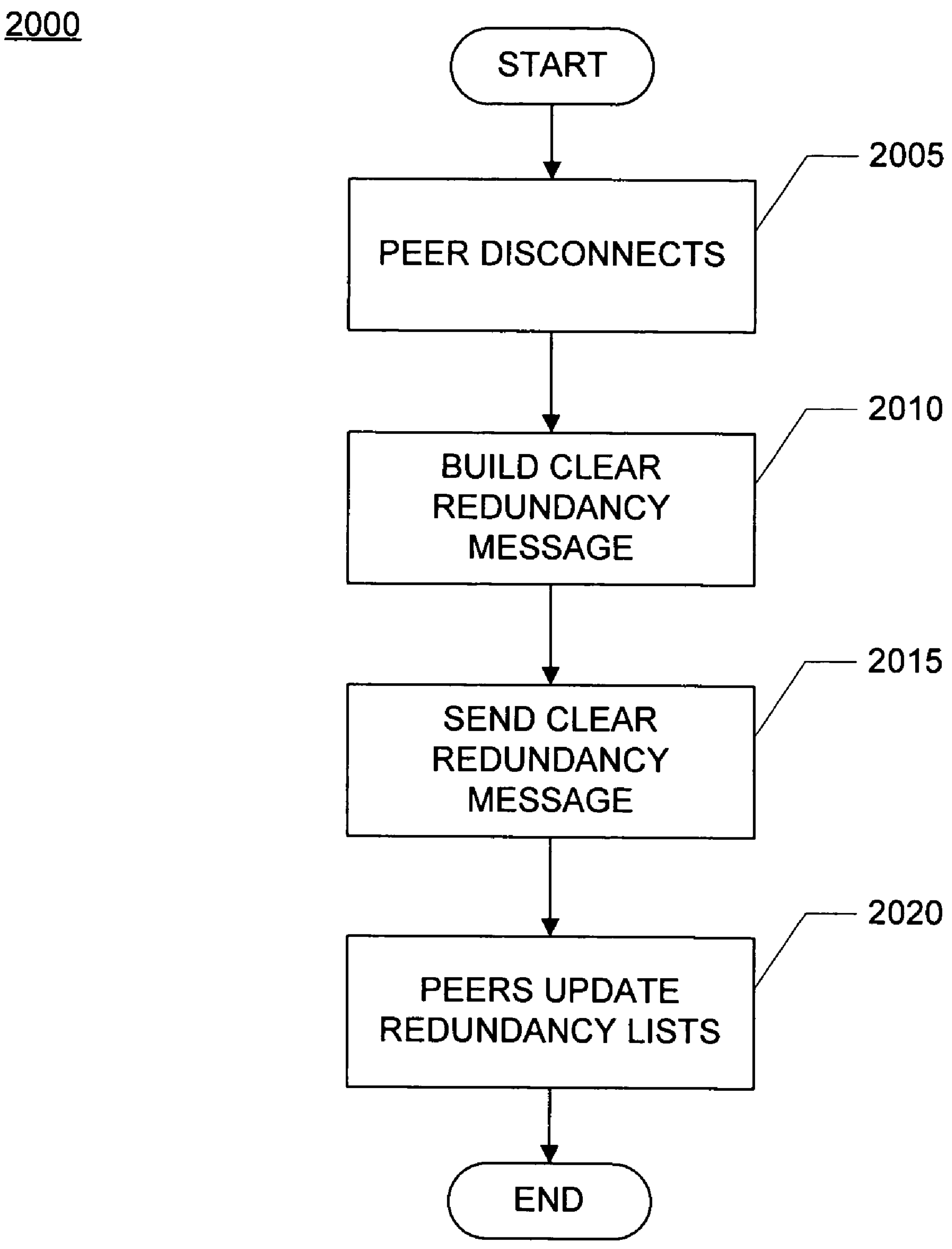

2000
START
PEER DISCONNECTS
2005
BUILD CLEAR REDUNDANCY MESSAGE
2010
SEND CLEAR REDUNDANCY MESSAGE
2015
PEERS UPDATE REDUNDANCY LISTS
2020
END

FIG. 21

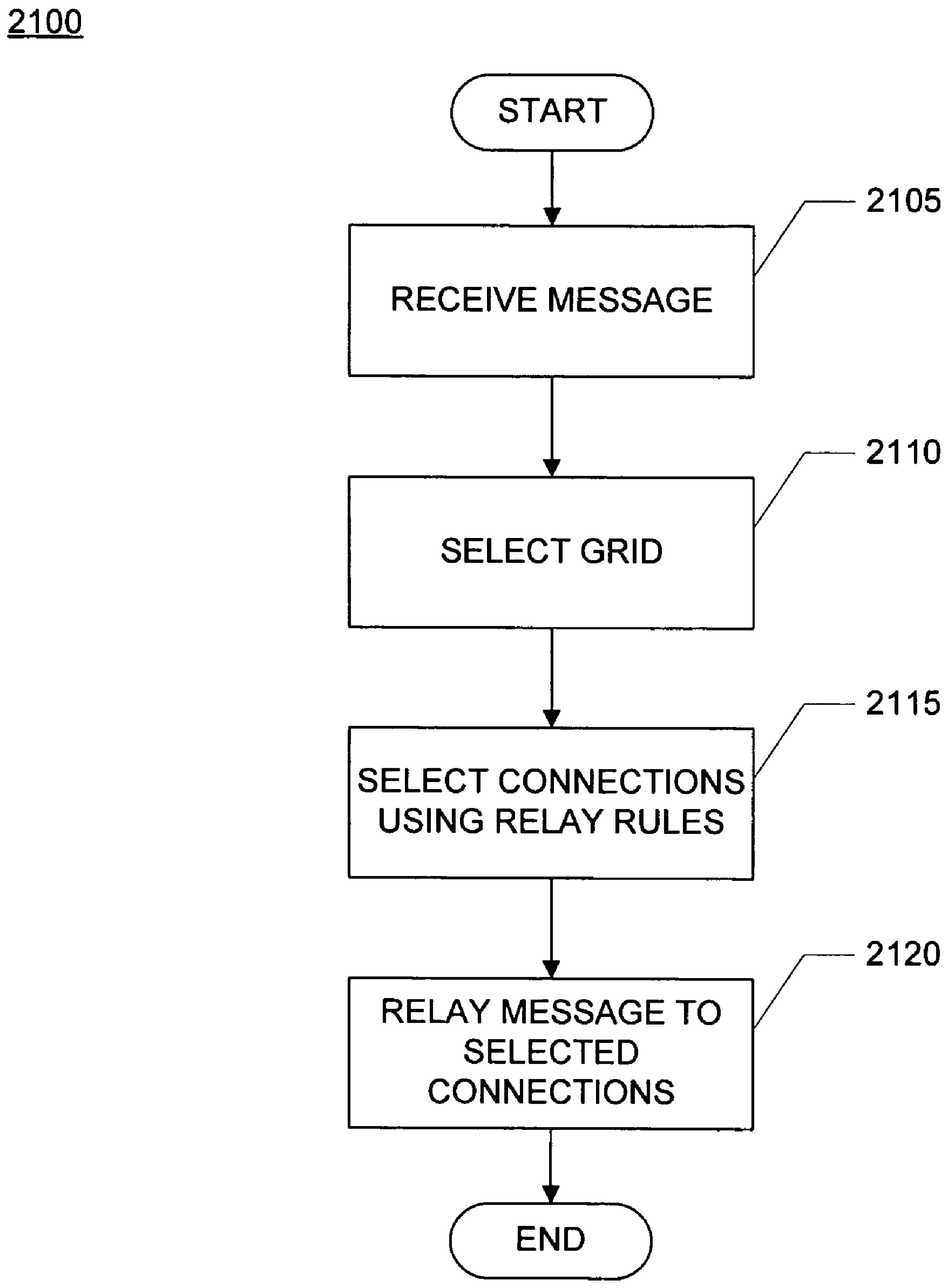

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

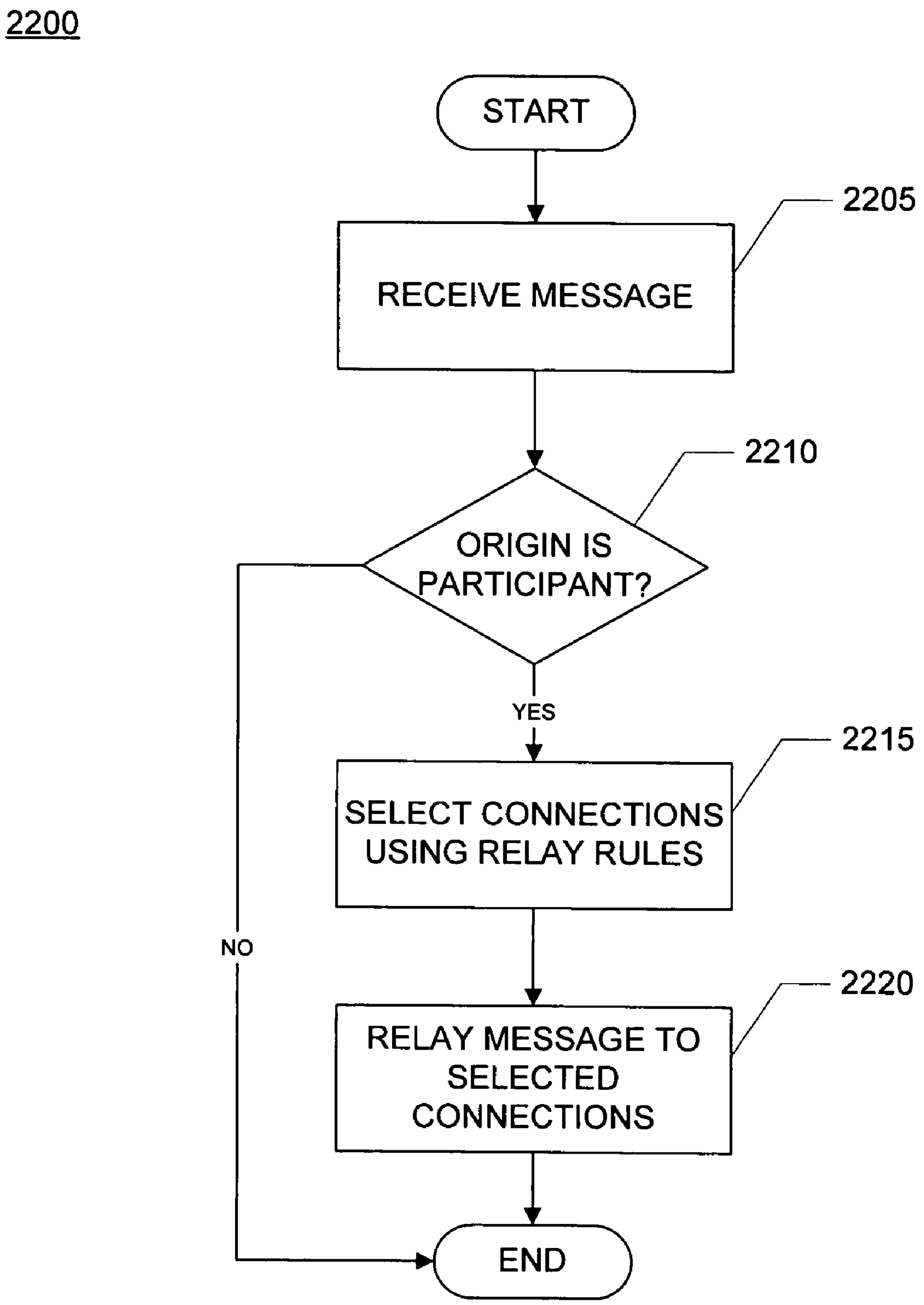

2200
START
RECEIVE MESSAGE
2205
ORIGIN IS PARTICIPANT?
2210
YES
NO
SELECT CONNECTIONS USING RELAY RULES
2215
RELAY MESSAGE TO SELECTED CONNECTIONS
2220
END

FIG. 23

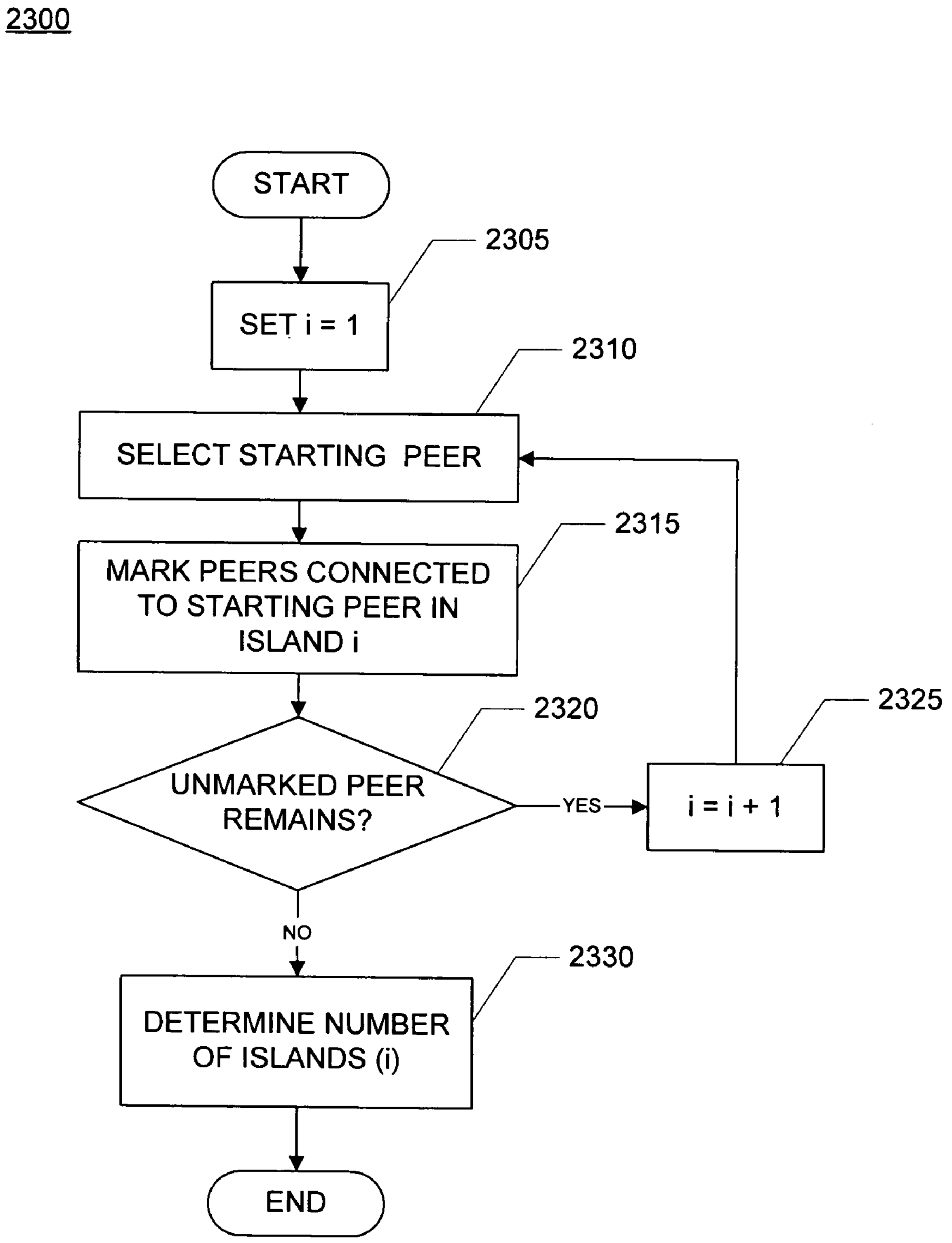

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

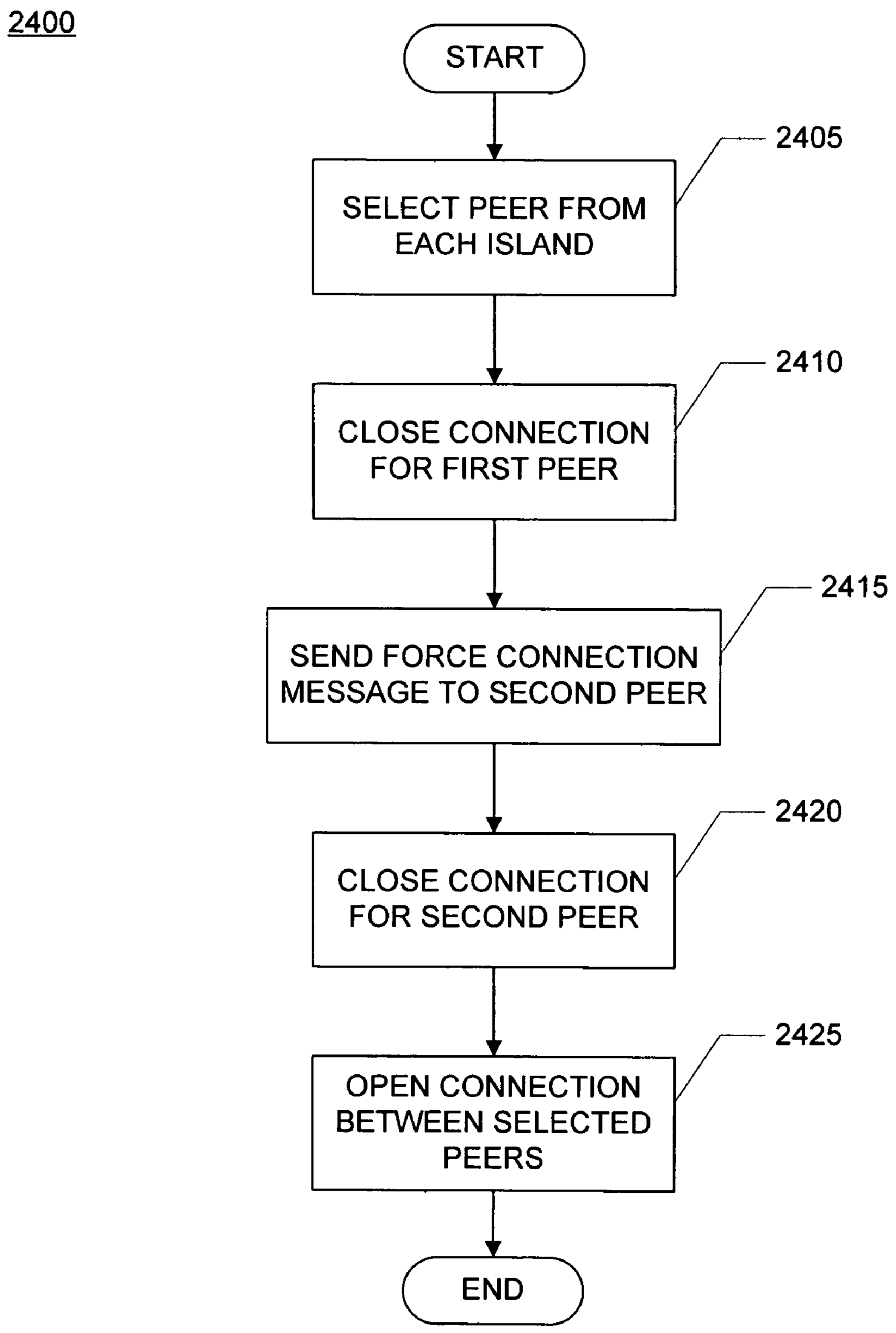

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

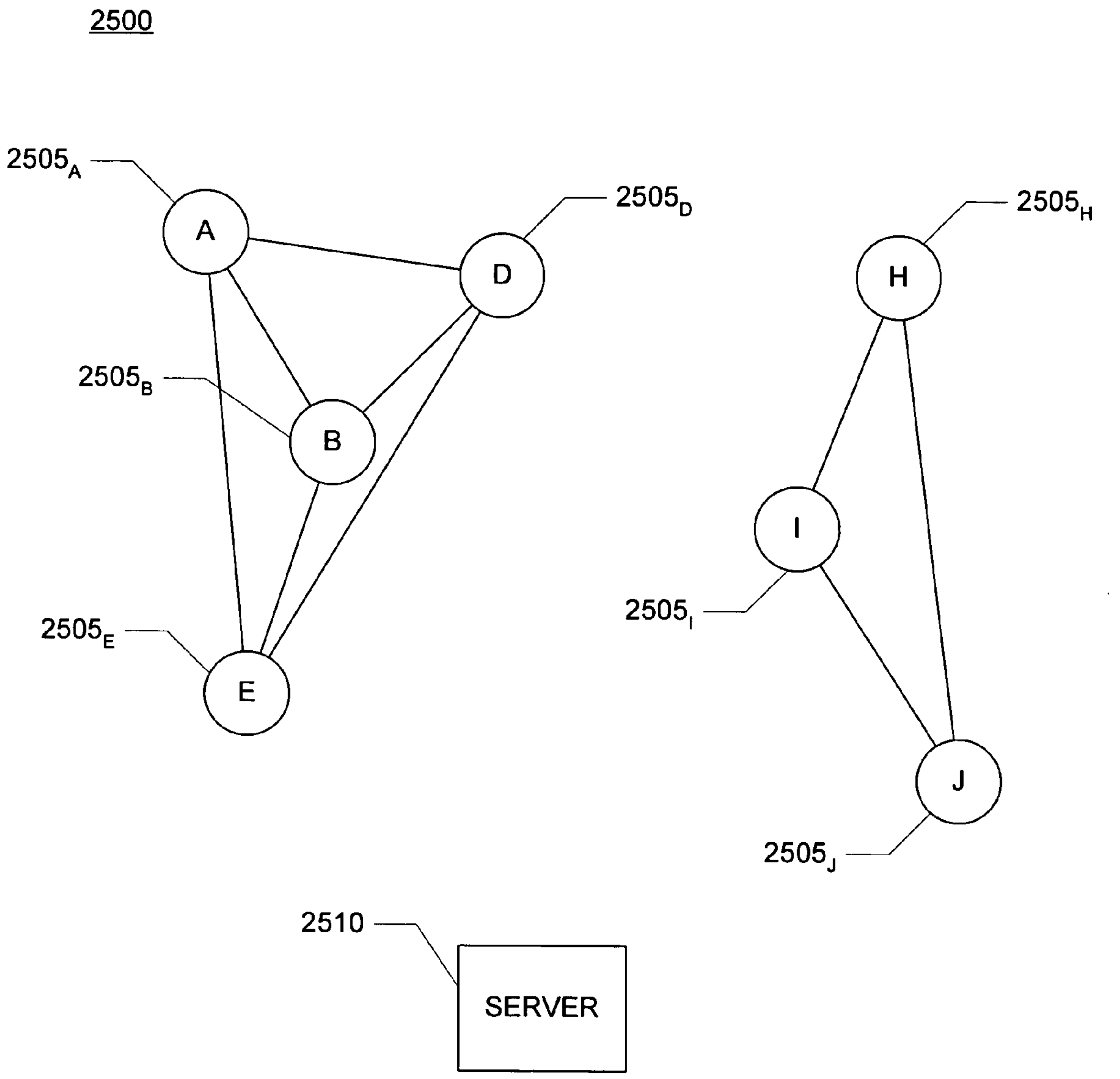

2500
2505_A
A
2505_D
D
2505_H
H
2505_B
B
I
2505_I
2505_E
E
J
2505_J
2510
SERVER

FIG. 26

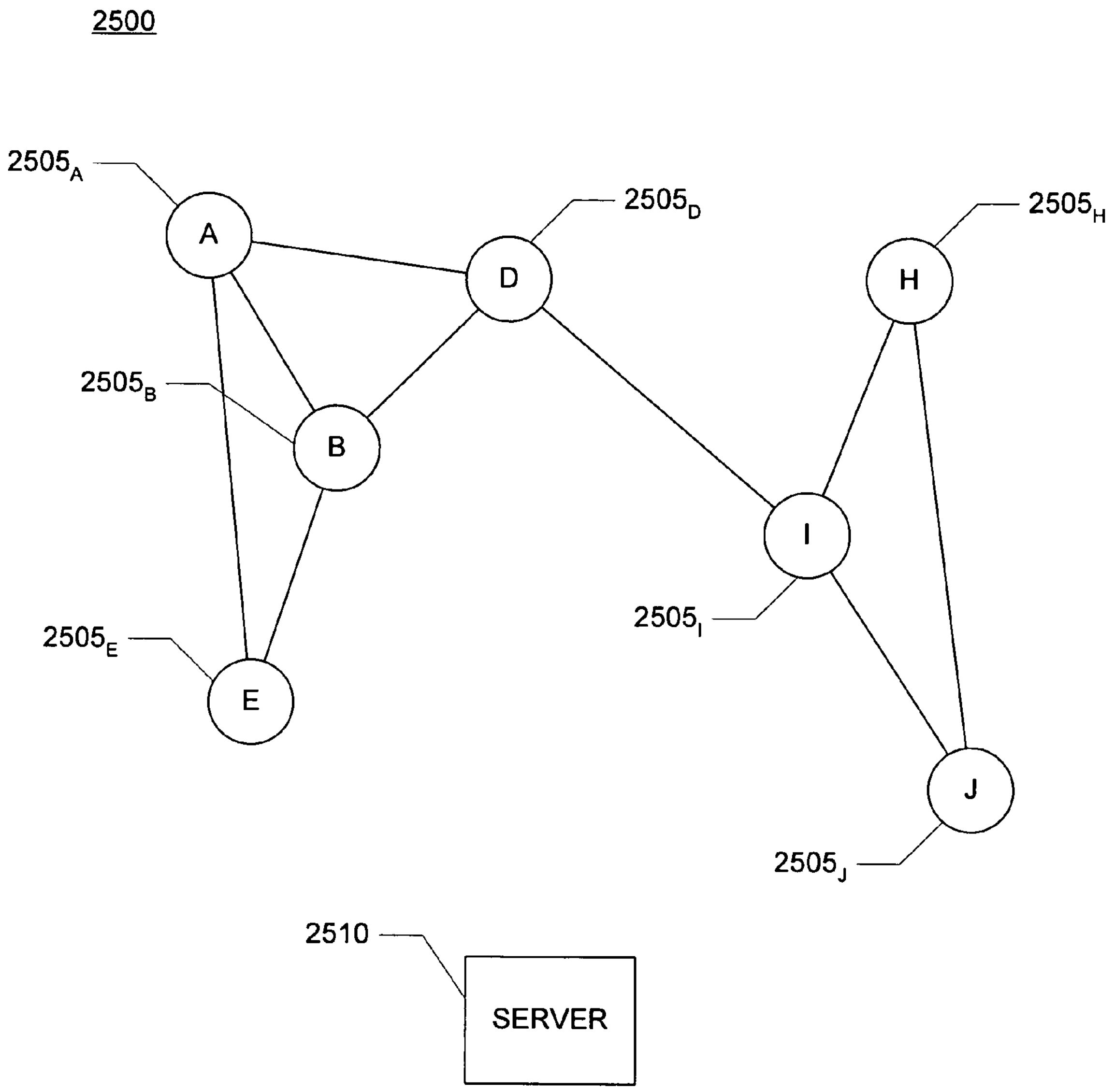

2500
2505_A
A
2505_D
D
2505_H
H
2505_B
B
I
2505_I
2505_E
E
J
2505_J
2510
SERVER

FIG. 27

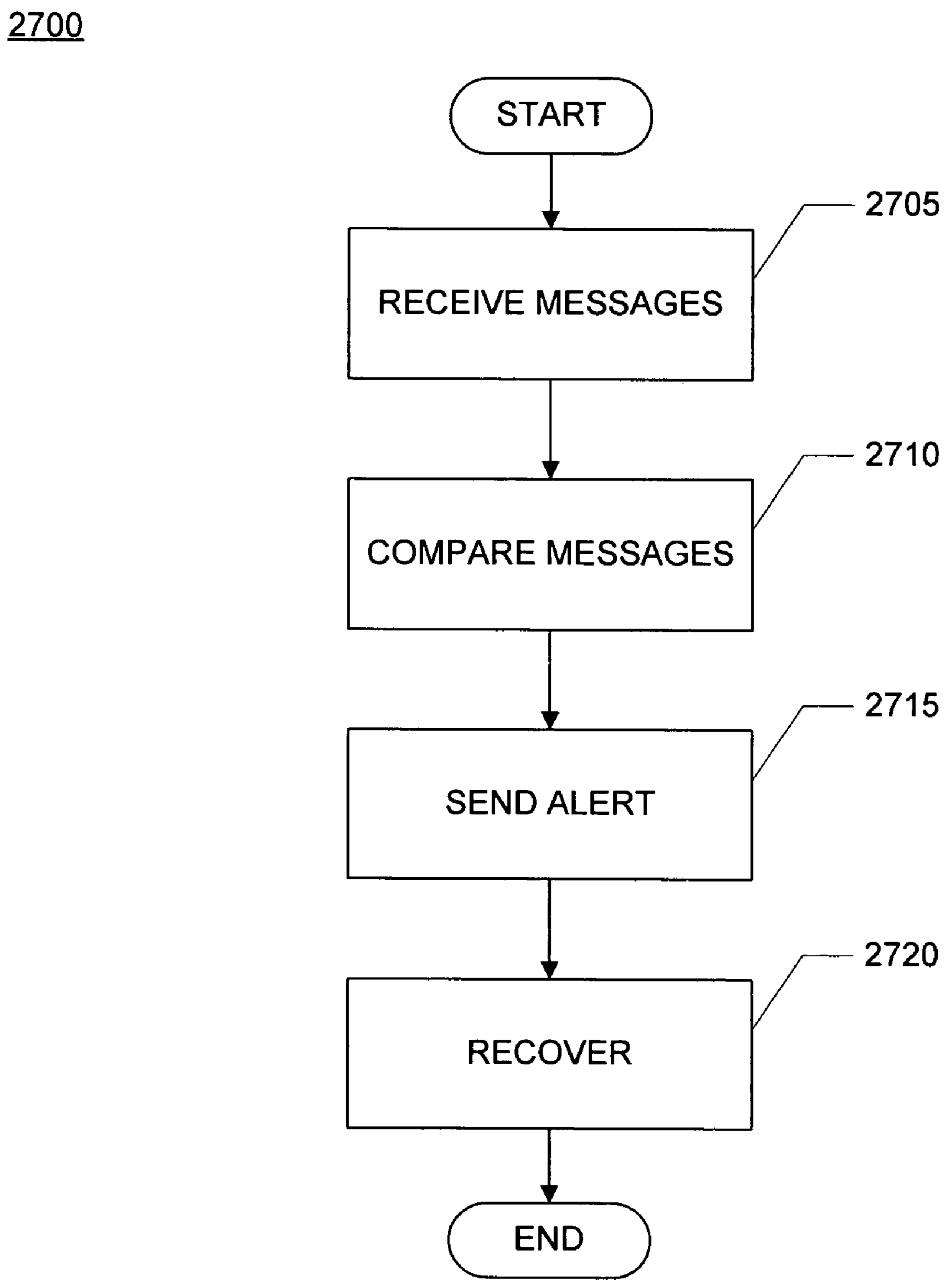

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

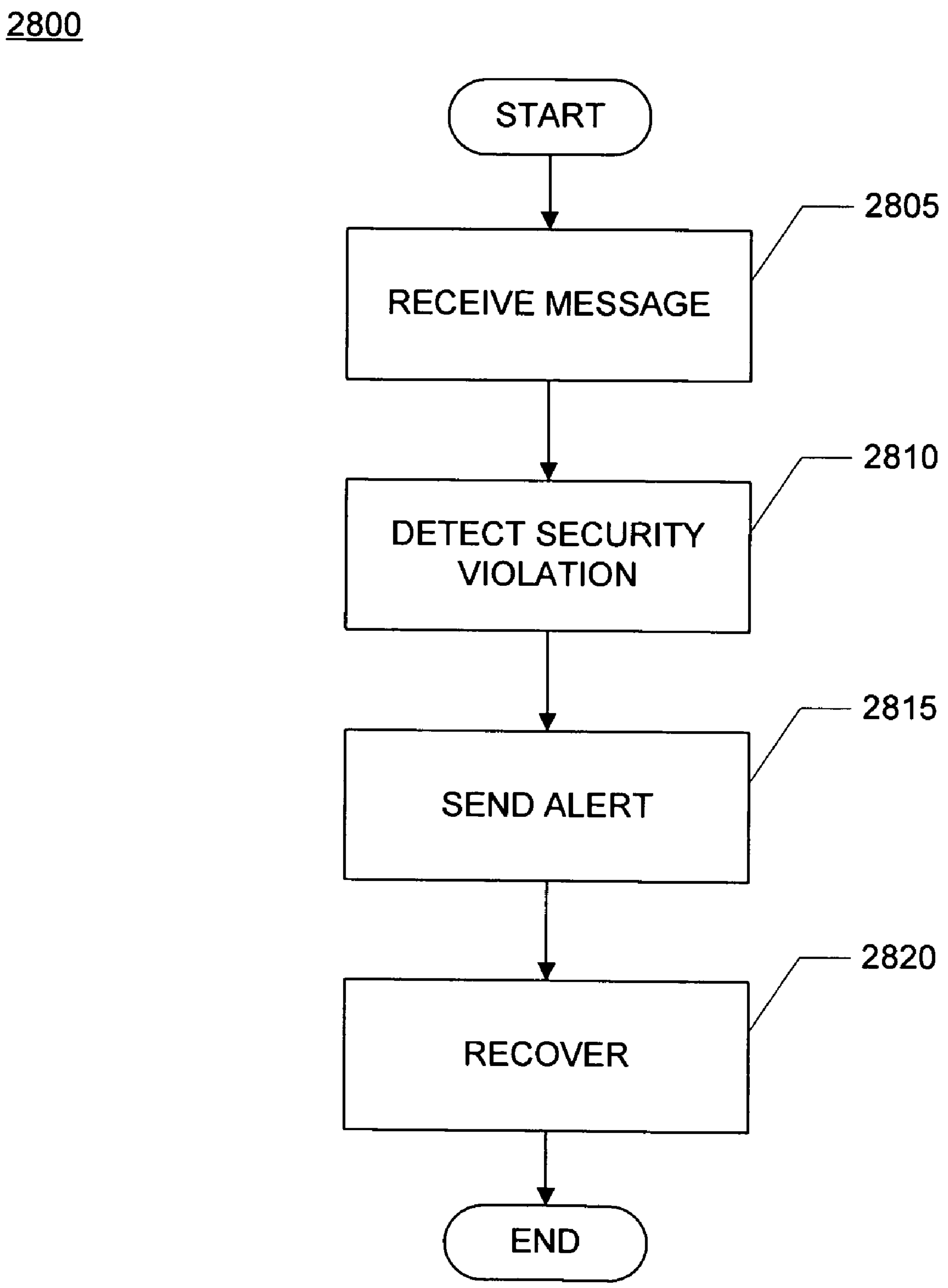

2800
START
2805
RECEIVE MESSAGE
2810
DETECT SECURITY VIOLATION
2815
SEND ALERT
2820
RECOVER
END

FIG. 29

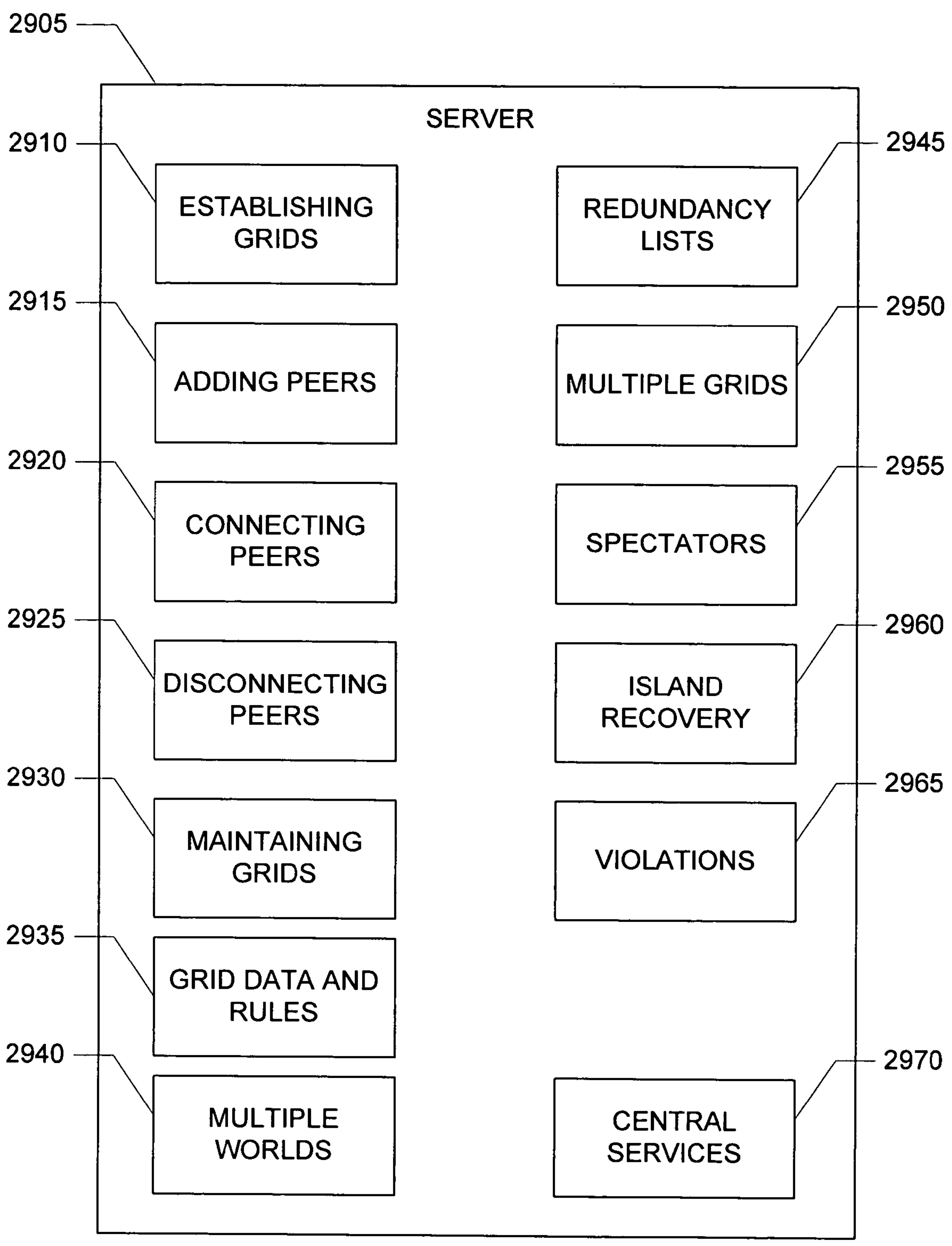

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

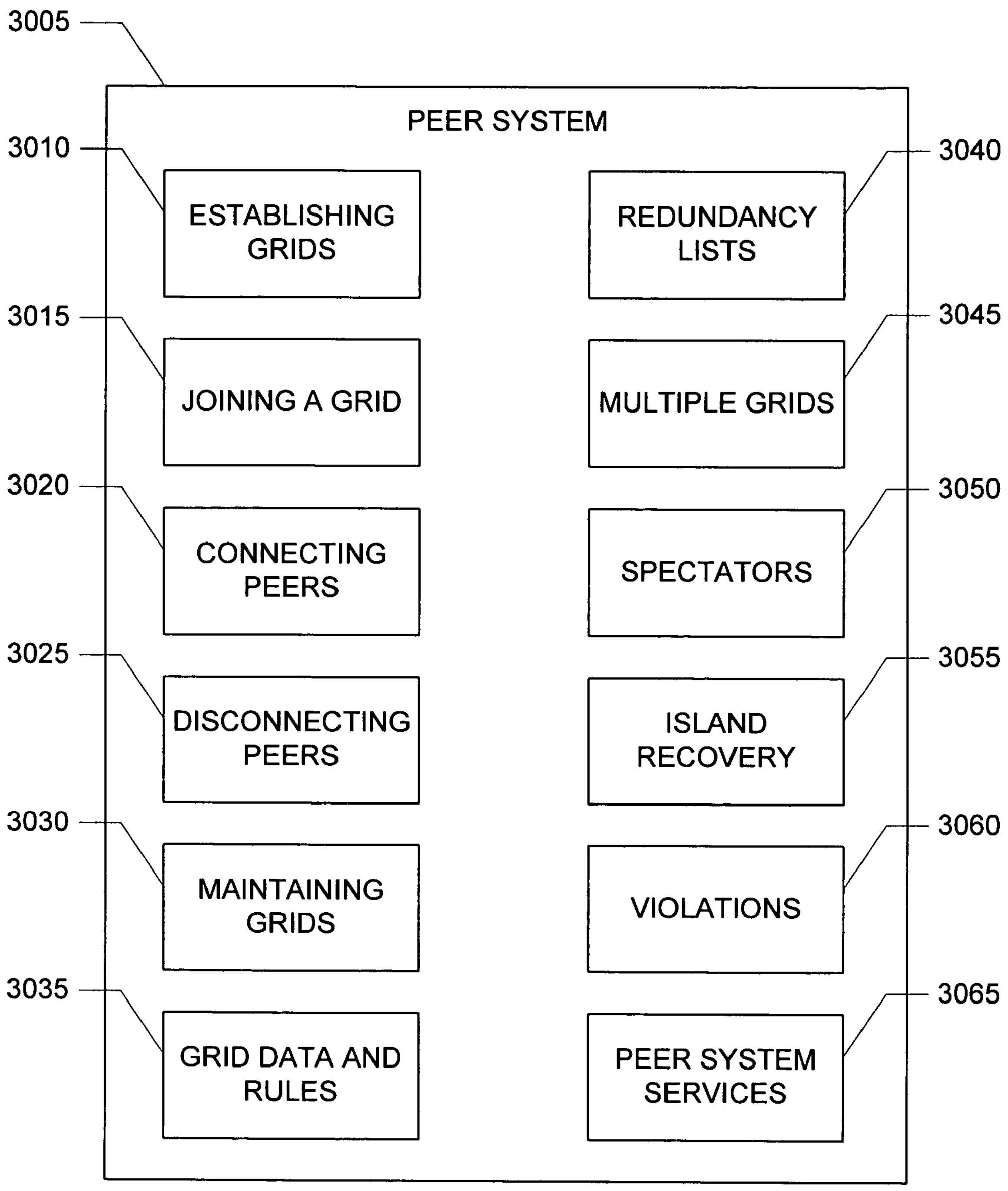

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

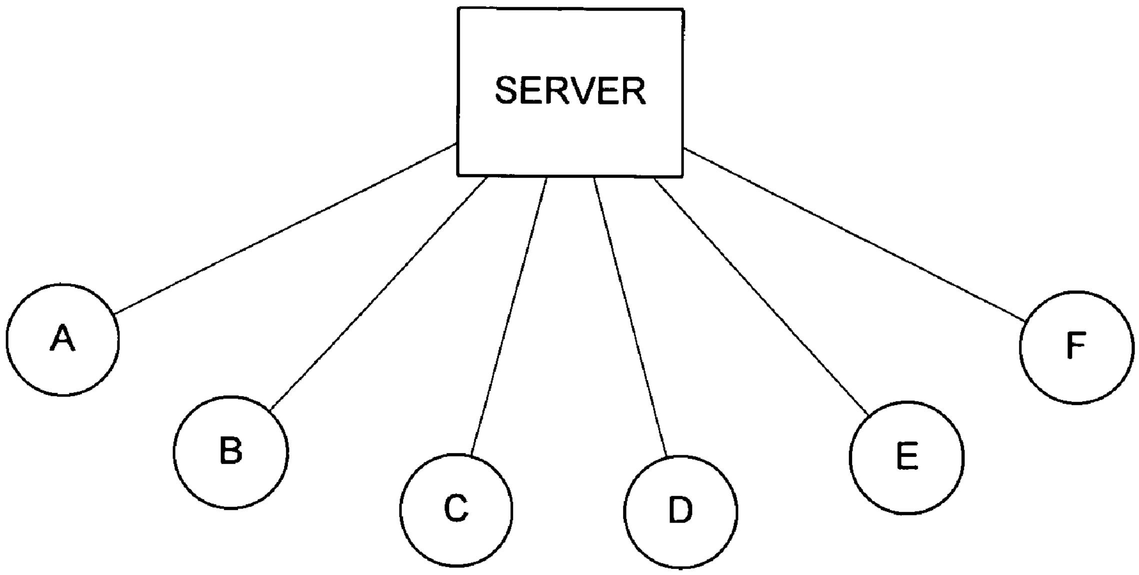
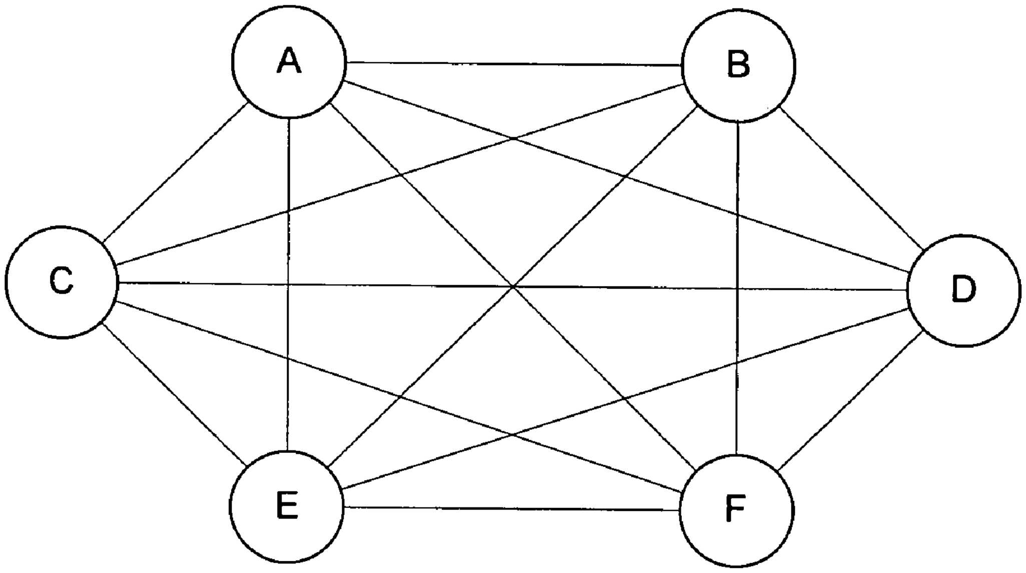


FIG. 31B



VIOLATIONS IN A PEER-TO-PEER RELAY NETWORK

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

This application is related to the U.S. applications Ser. No. 10/700,798, filed on Nov. 3, 2003, Ser. No. 10/701,302, filed on Nov. 3, 2003, Ser. No. 10/701,014, filed on Nov. 3, 2003, Ser. No. 10/700,777, filed on Nov. 3, 2003, and Ser. No. 10/701,298, filed on Nov. 3, 2003.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N−1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a method of detecting and recovering from violations in a peer-to-peer relay network includes: receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network detecting a violation in said received message; and sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

In one implementation, a peer system in a peer-to-peer relay network includes: means for receiving a message at a peer system from a sending peer system connected to said peer system in a peer-to-peer relay network; means for detecting a violation in said received message; and means for sending an alert message to each peer system connected to said peer system in said peer-to-peer relay network; wherein each peer system in said peer-to-peer relay network stores a connection limit defining a number of other peer systems up to which that peer system is permitted to connect, and each peer system stores a set of one or more relay rules for relaying data to other peer systems connected to that peer system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent applications Ser. Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed Jul. 31, 2002, and 10/359,359, ("Multi-User Application Programming Interface"), filed Feb. 4, 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system **105**$_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer

relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins, peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available

connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that

receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer

stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method of detecting and recovering from a violation in a peer-to-peer relay network, comprising:

sending a first message having first content data from a sending peer system to a receiving peer system;

detecting a manipulation of data in said sent first message, said manipulation of data changing the outcome of processing by the receiving peer system;

wherein detecting said manipulation includes:

relaying back the sent first message to the sending peer by the receiving peer system;

comparing by the sending peer of the relayed back message to the sent first message to identify a receiving peer responsible for the manipulation of data; and

sending a manipulated data alert message to other peer systems connected to the peer system in said peer-to-peer relay network, the manipulated data alert message identifying the receiving peer as responsible for the manipulation of data.

2. The method of claim **1**, further comprising ignoring messages from the sending peer responsible for the manipulation of data.

3. The method of claim **1**, further comprising forcing the sending peer responsible for the manipulation of data to disconnect from the peer-to-peer relay network.

* * * * *

EXHIBIT 87

US007627678B2

(12) **United States Patent**
Datta et al.

(10) **Patent No.: US 7,627,678 B2**
(45) **Date of Patent: Dec. 1, 2009**

(54) CONNECTING A PEER IN A PEER-TO-PEER RELAY NETWORK

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Foster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 805 days.

(21) Appl. No.: **10/700,798**

(22) Filed: **Nov. 3, 2003**

(65) **Prior Publication Data**

US 2005/0086288 A1 Apr. 21, 2005

Related U.S. Application Data

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/227**; 709/204; 709/223; 709/225; 709/238; 709/243

(58) **Field of Classification Search** 709/223, 709/225, 227, 238, 204, 243
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,269,099	B1 *	7/2001	Borella et al.	370/389
6,421,347	B1 *	7/2002	Borgstahl et al.	370/310
6,434,543	B1 *	8/2002	Goldberg et al.	707/2
7,065,579	B2 *	6/2006	Traversat et al.	709/225
7,130,921	B2 *	10/2006	Goodman et al.	709/238
7,174,382	B2 *	2/2007	Ramanathan et al.	709/227
7,263,070	B1 *	8/2007	Delker et al.	709/227
7,272,636	B2 *	9/2007	Pabla	709/223
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0062375	A1 *	5/2002	Teodosiu et al.	709/226
2002/0073204	A1 *	6/2002	Dutta et al.	709/227

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999
EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

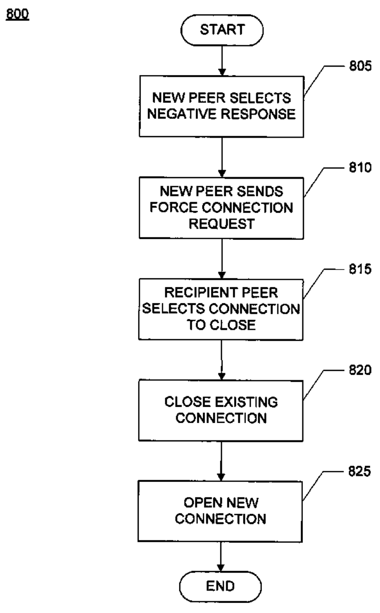
(Continued)

Primary Examiner—Ramy M Osman
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, and each peer system is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules.

19 Claims, 31 Drawing Sheets


800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

U.S. PATENT DOCUMENTS

2002/0119821	A1	8/2002	Sen et al.	
2002/0143855	A1 *	10/2002	Traversat et al.	709/202
2002/0156875	A1 *	10/2002	Pabla	709/220
2002/0161821	A1 *	10/2002	Narayan et al.	709/204
2002/0184310	A1	12/2002	Traversat et al.	
2003/0055892	A1	3/2003	Huitema et al.	
2003/0120662	A1 *	6/2003	Vishik	707/100
2003/0126245	A1 *	7/2003	Feltin et al.	709/223

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrail : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinkly, Finland, Jun. 11-14, 2001, IEEE International Conference on Communication, New York, NY: IEEE, US, vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault—Tolerant key Agreement by Dynamic Collaborative Groups" Proceedings of the 7th ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. Conf. 7, Nov. 1, 2000, pp. 1-38, XP 0029551317 ISBN: 1-58113-203-4.

* cited by examiner

FIG. 1

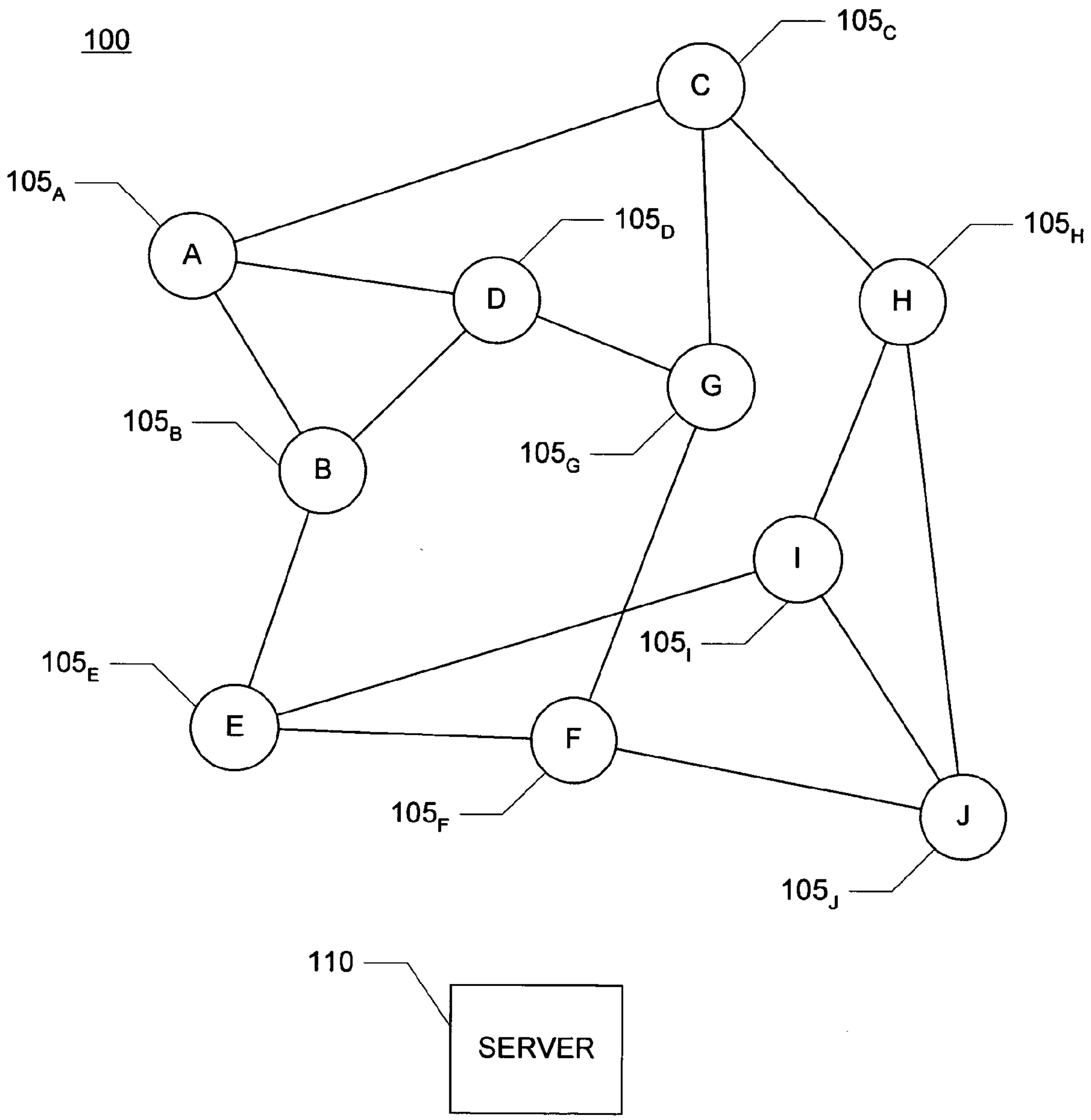

100
105$_C$
C
105$_A$
A
105$_D$
D
105$_H$
H
G
105$_B$
B
105$_G$
I
105$_I$
105$_E$
E
F
105$_F$
J
105$_J$
110
SERVER

FIG. 2

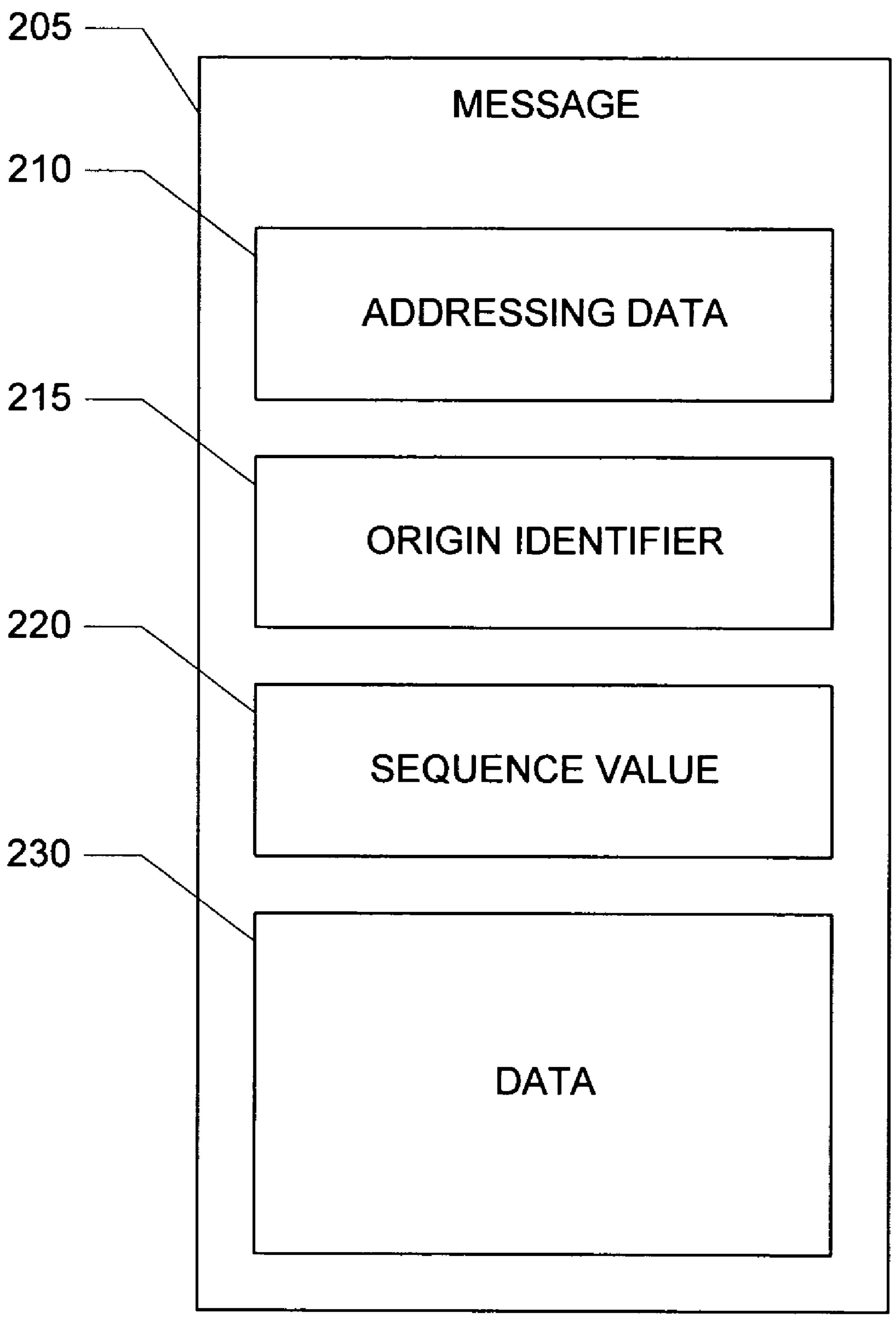

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

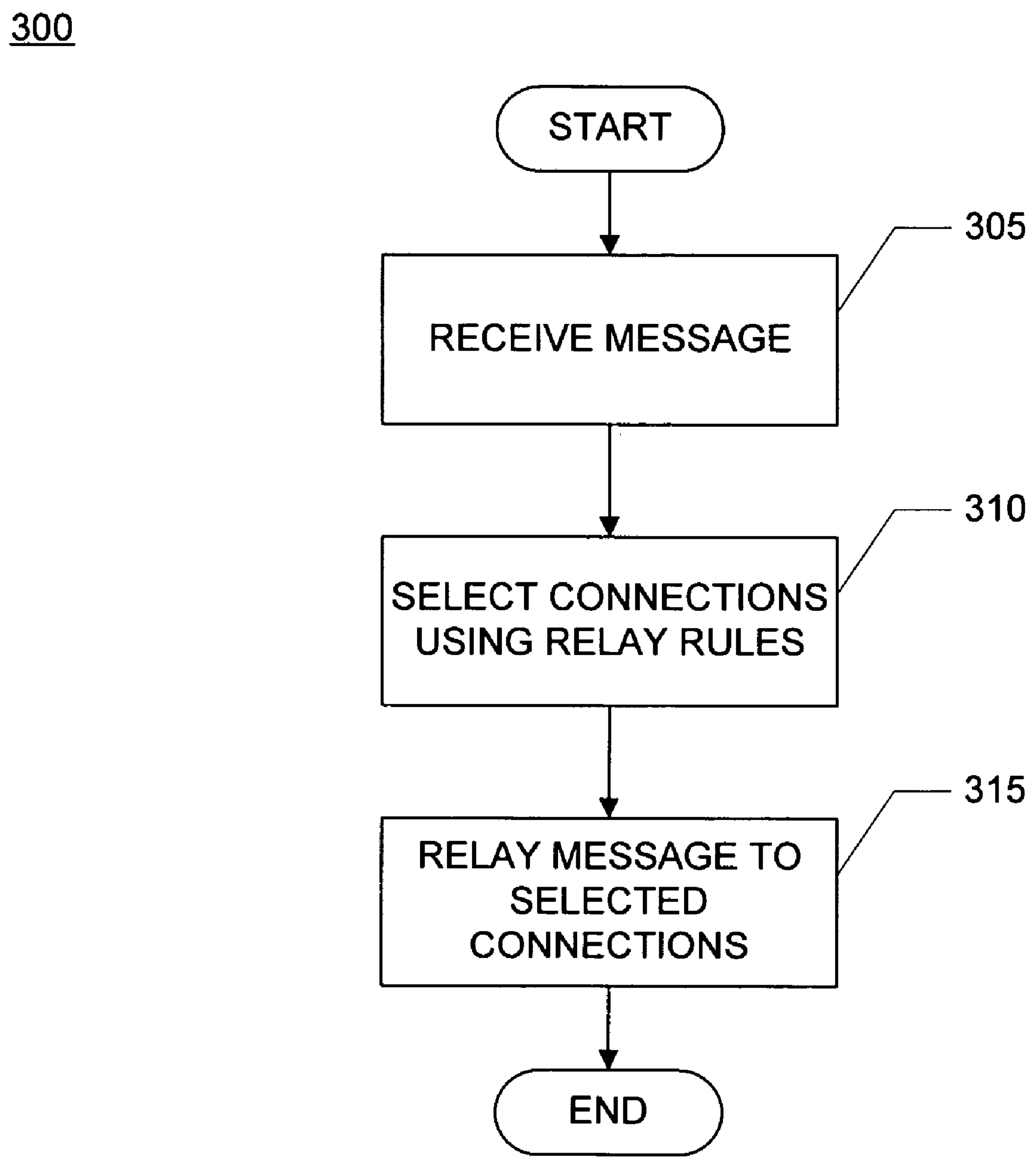

300
START
305
RECEIVE MESSAGE
310
SELECT CONNECTIONS USING RELAY RULES
315
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 4

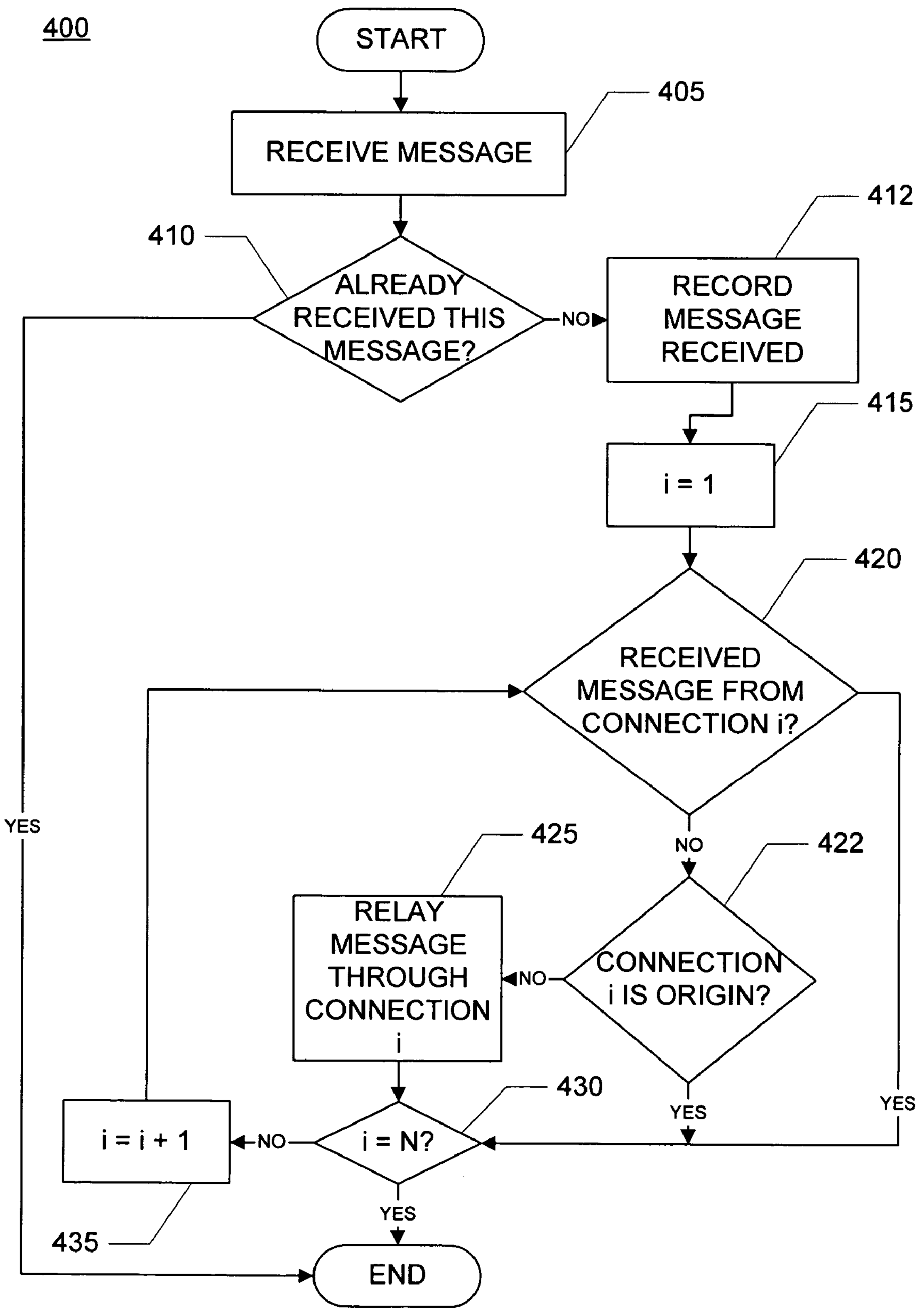

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
RECORD MESSAGE RECEIVED
412
415
i = 1
420
RECEIVED MESSAGE FROM CONNECTION i?
YES
NO
422
CONNECTION i IS ORIGIN?
NO
425
RELAY MESSAGE THROUGH CONNECTION i
YES
YES
430
i = N?
NO
i = i + 1
435
YES
END

FIG. 5

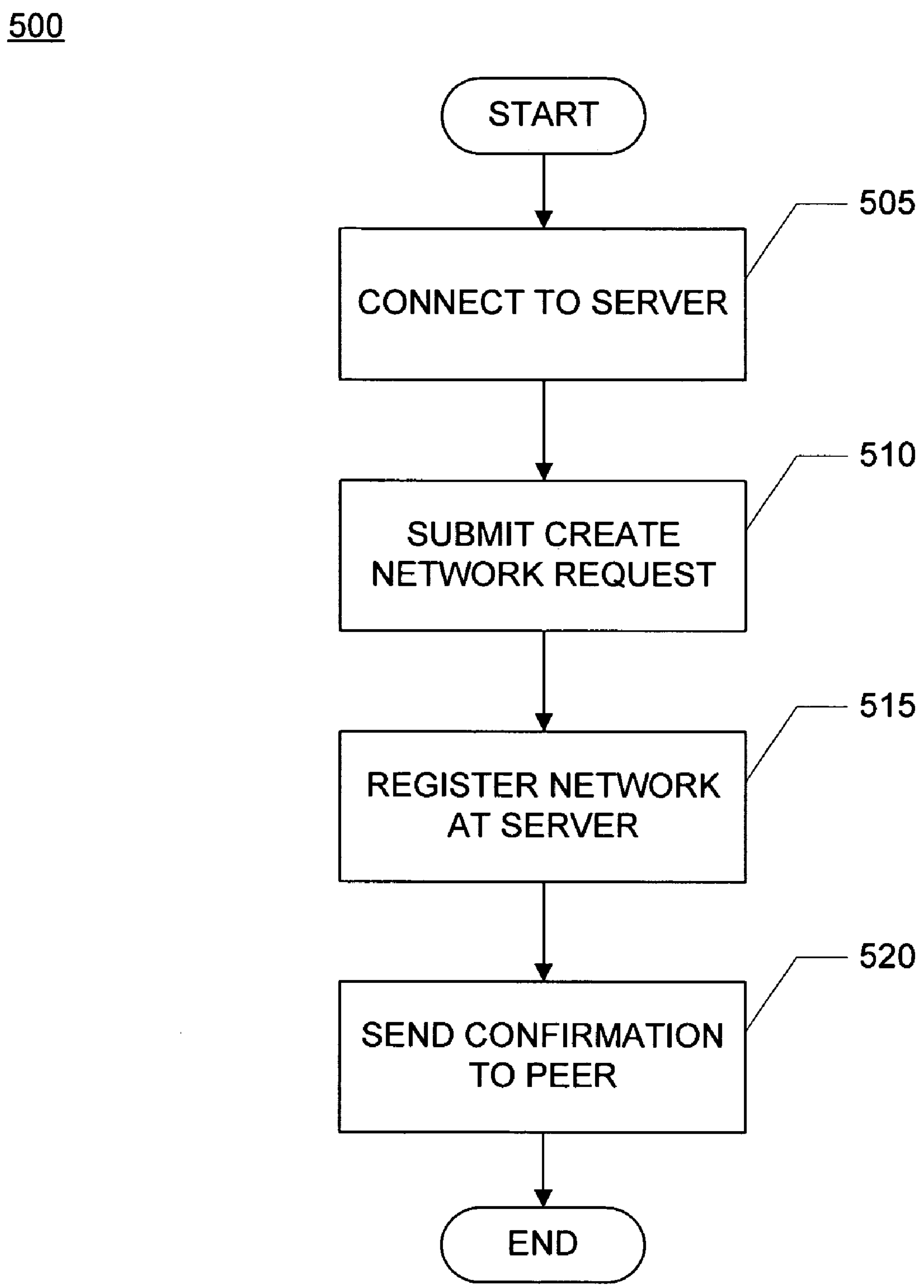

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

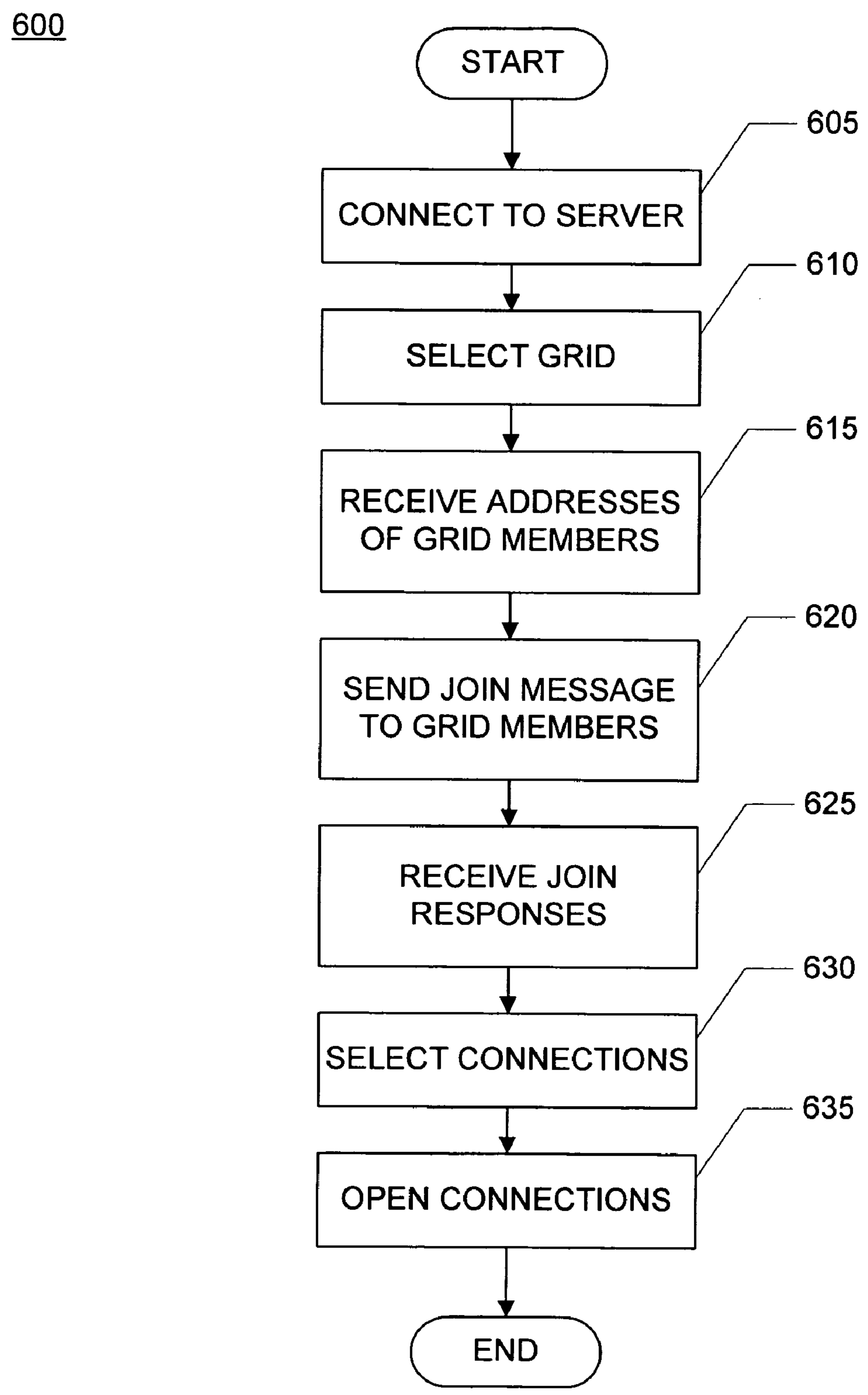

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

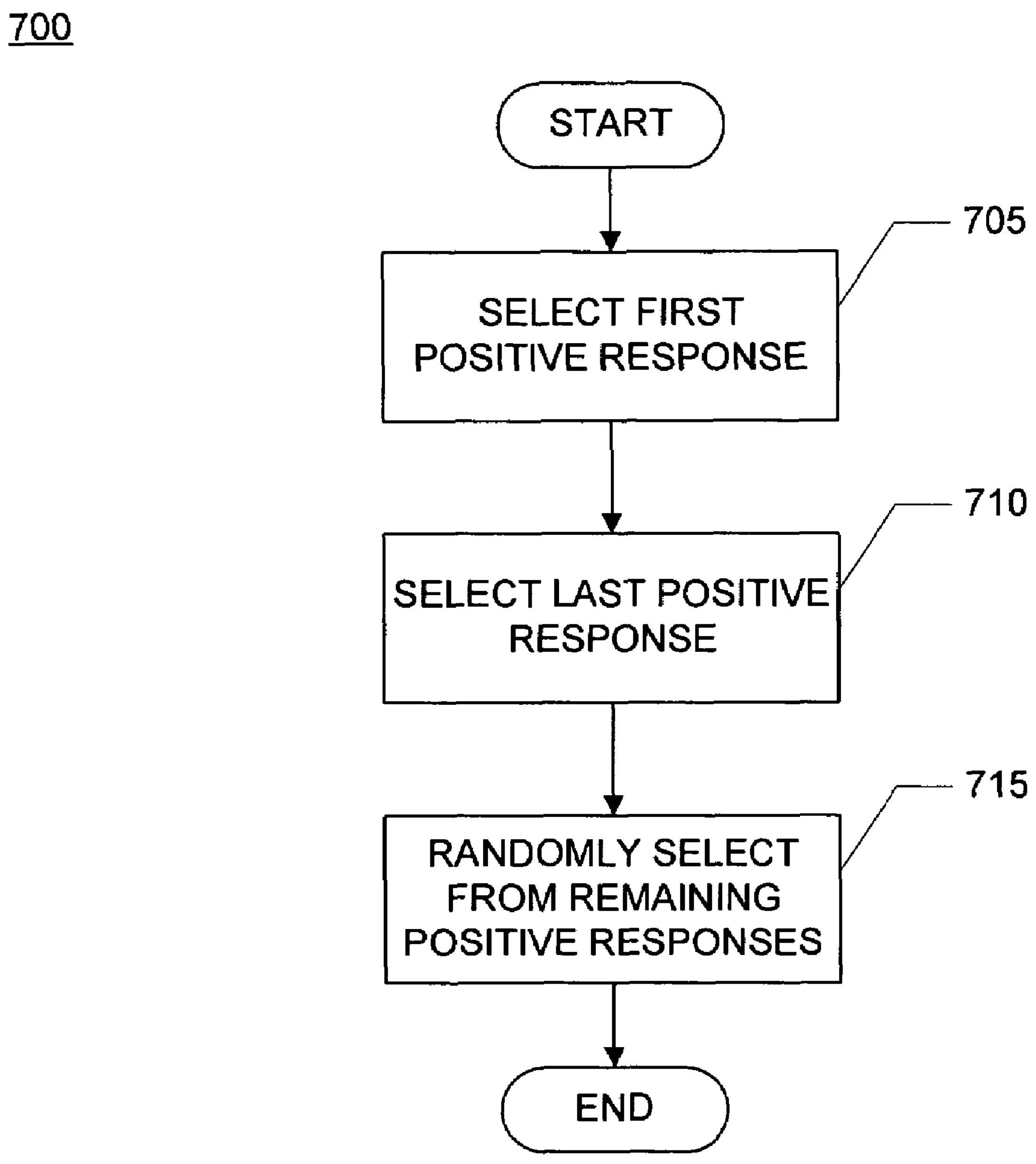

700
START
705
SELECT FIRST POSITIVE RESPONSE
710
SELECT LAST POSITIVE RESPONSE
715
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
END

FIG. 8

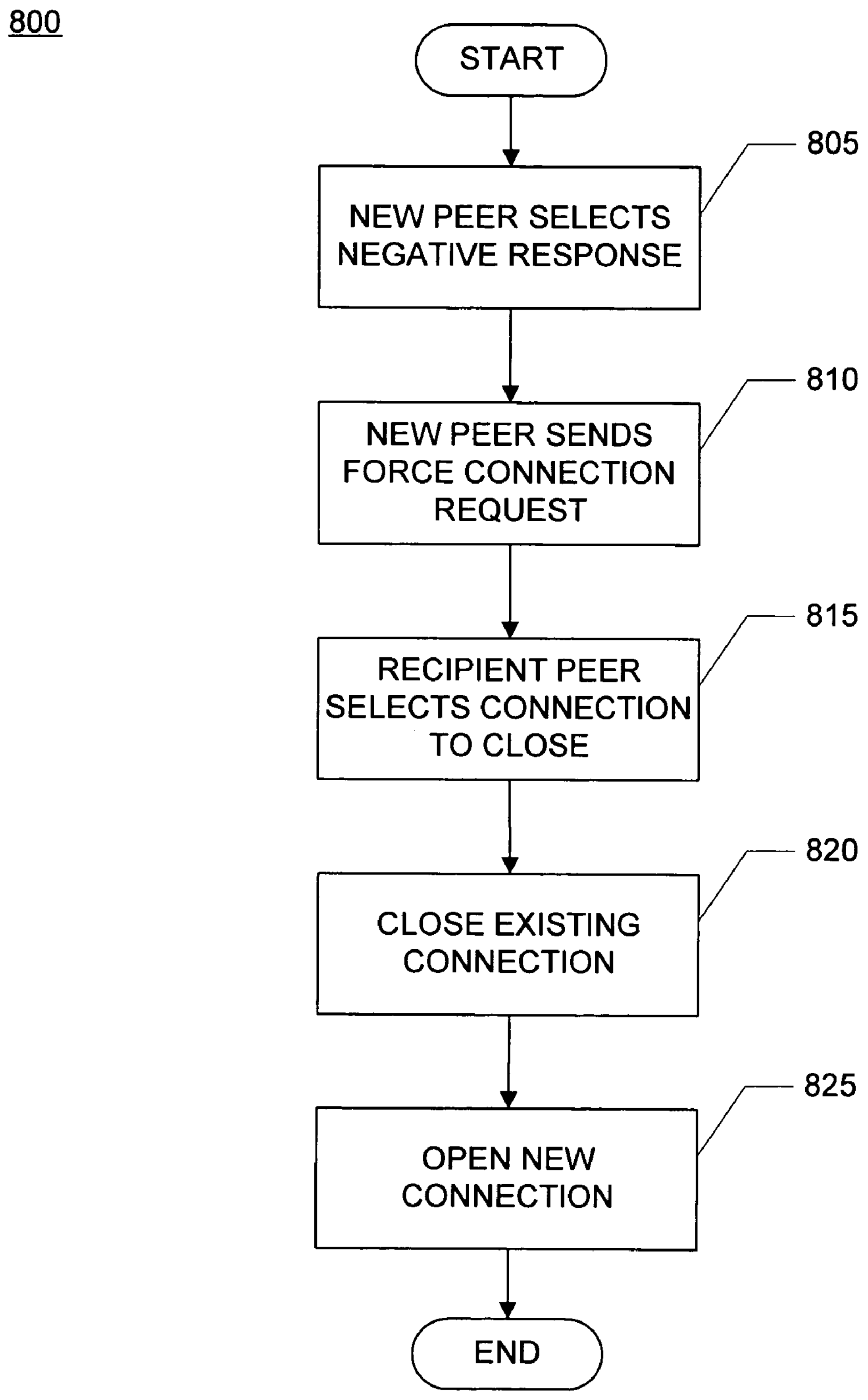

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

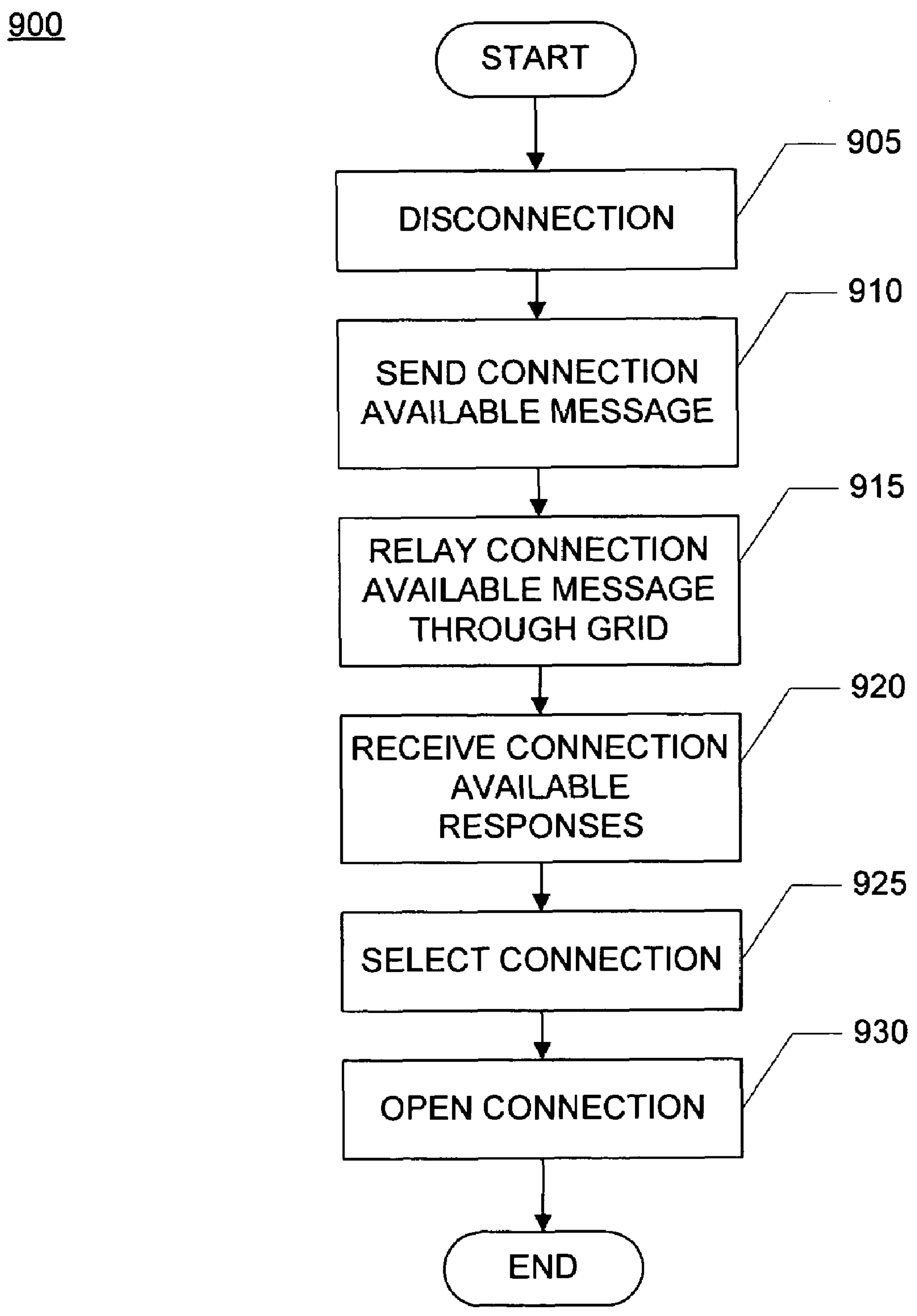

900
START
905
DISCONNECTION
910
SEND CONNECTION AVAILABLE MESSAGE
915
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
920
RECEIVE CONNECTION AVAILABLE RESPONSES
925
SELECT CONNECTION
930
OPEN CONNECTION
END

FIG. 10

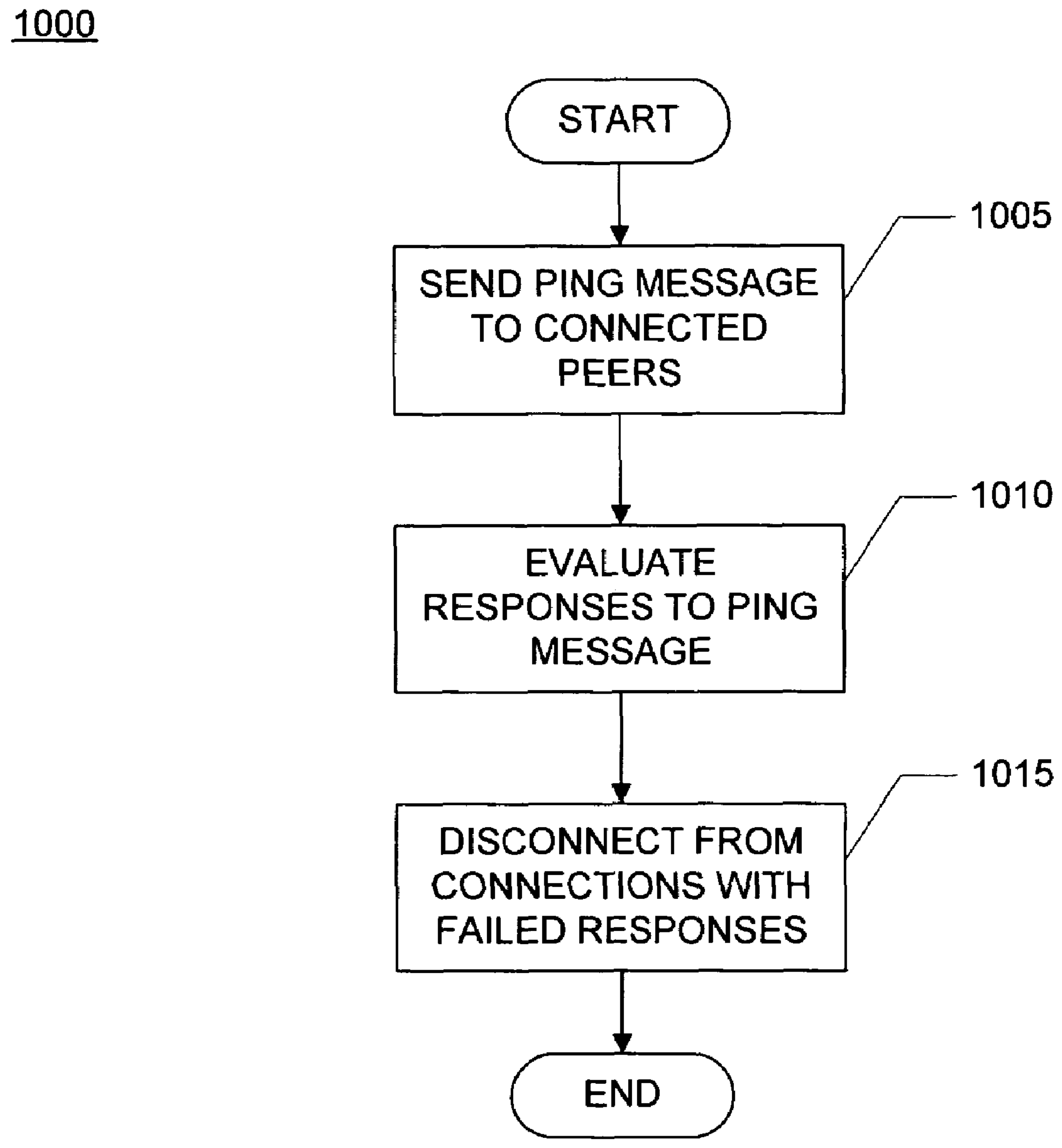

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

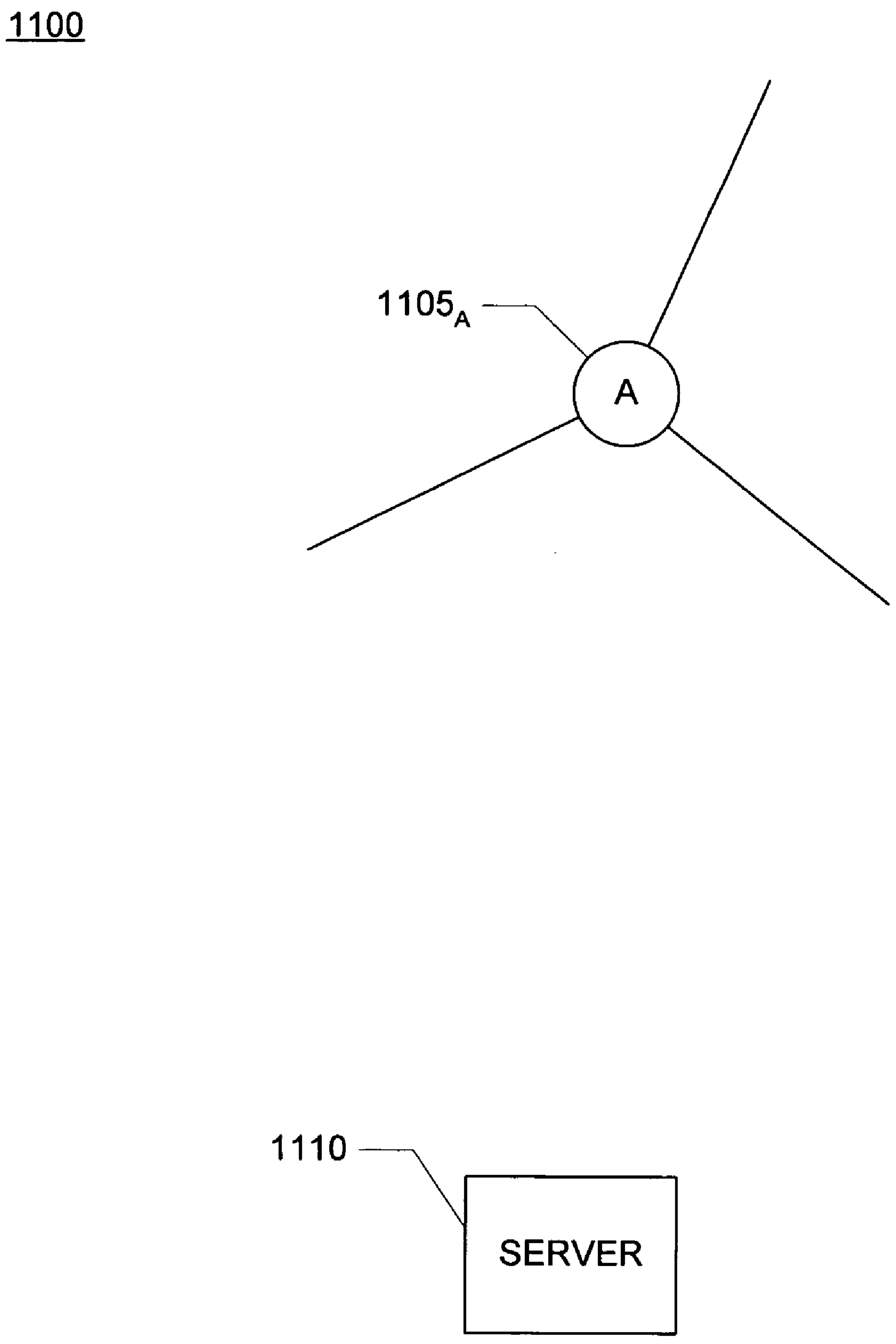


FIG. 12

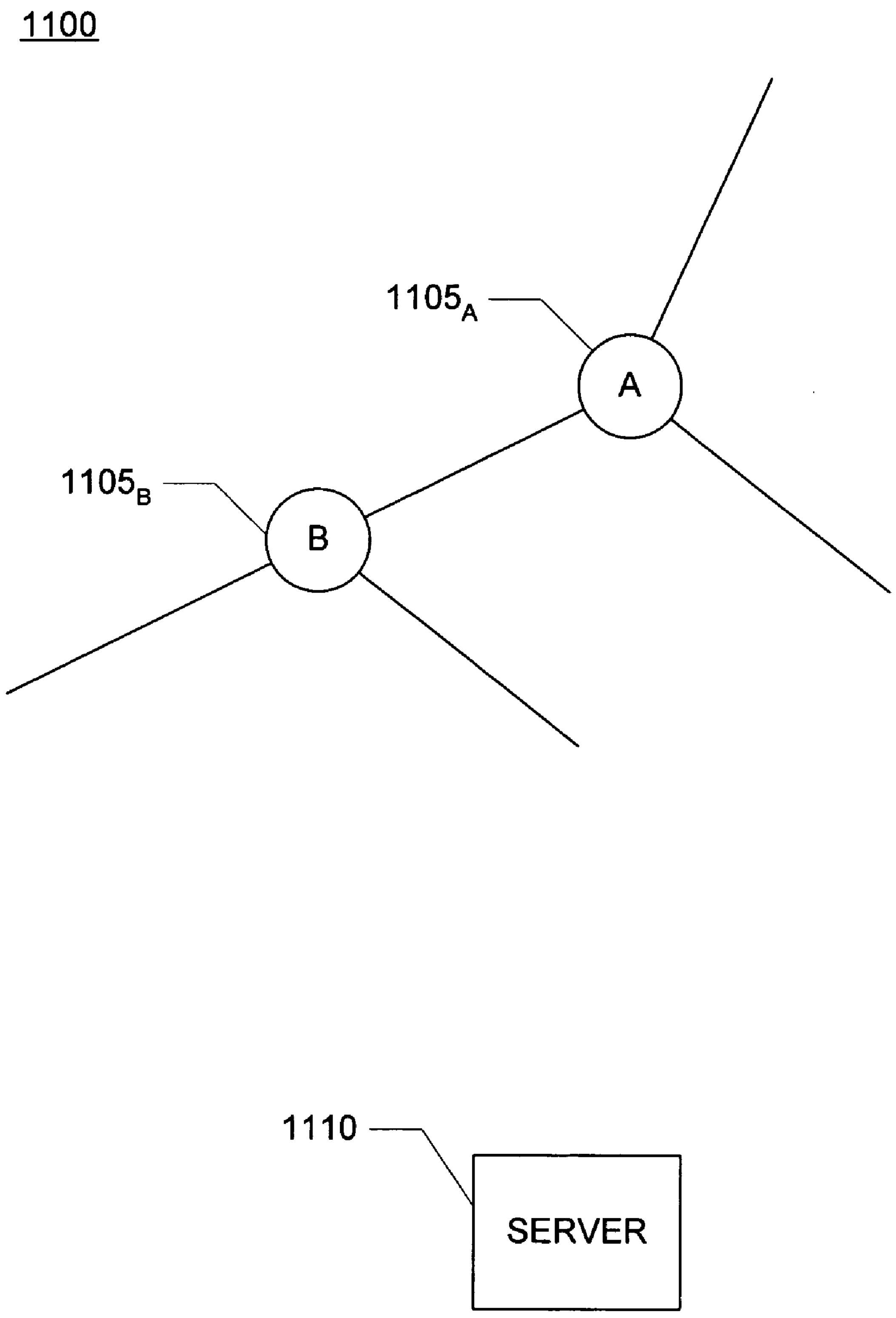


FIG. 13

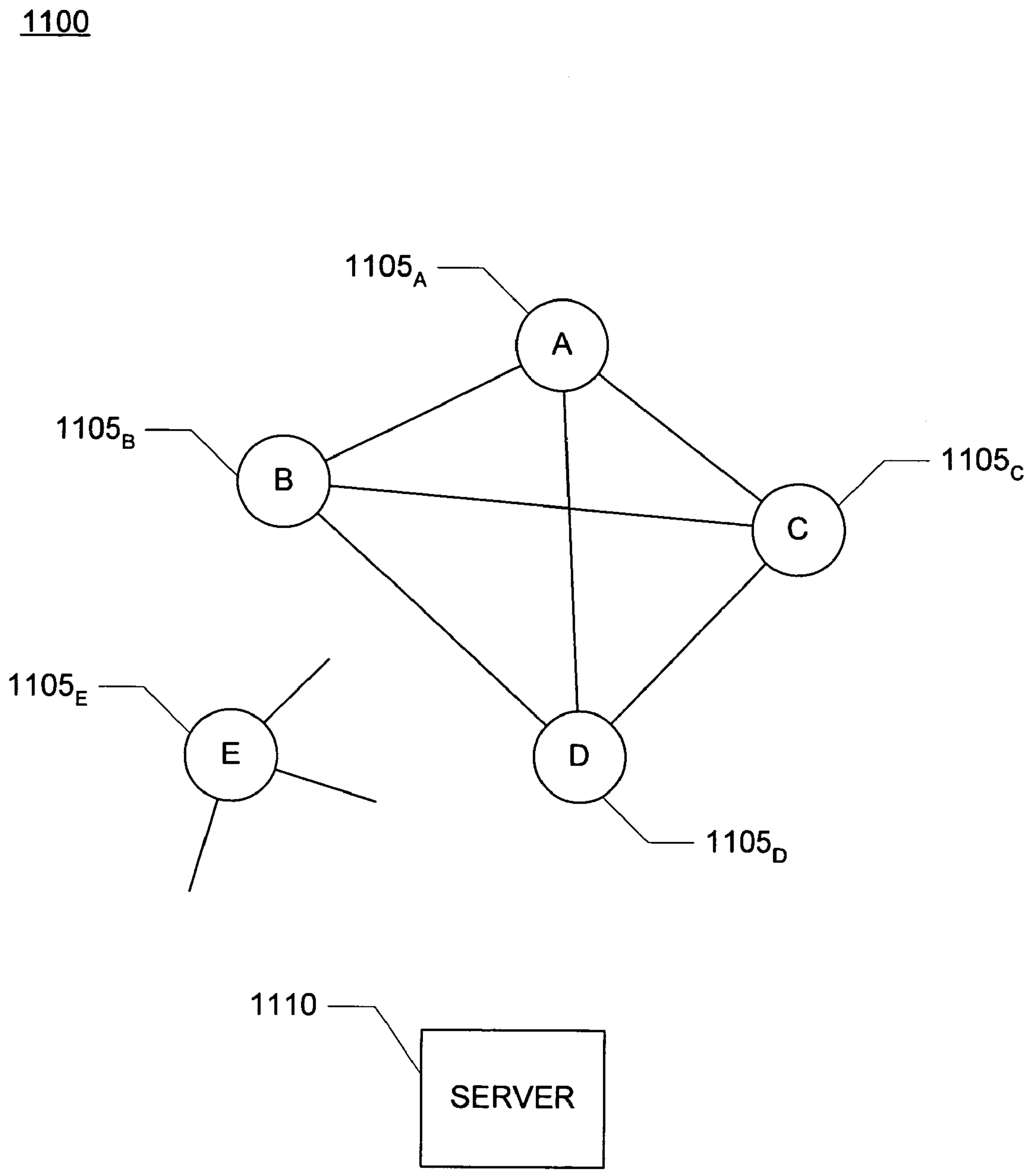


FIG. 14

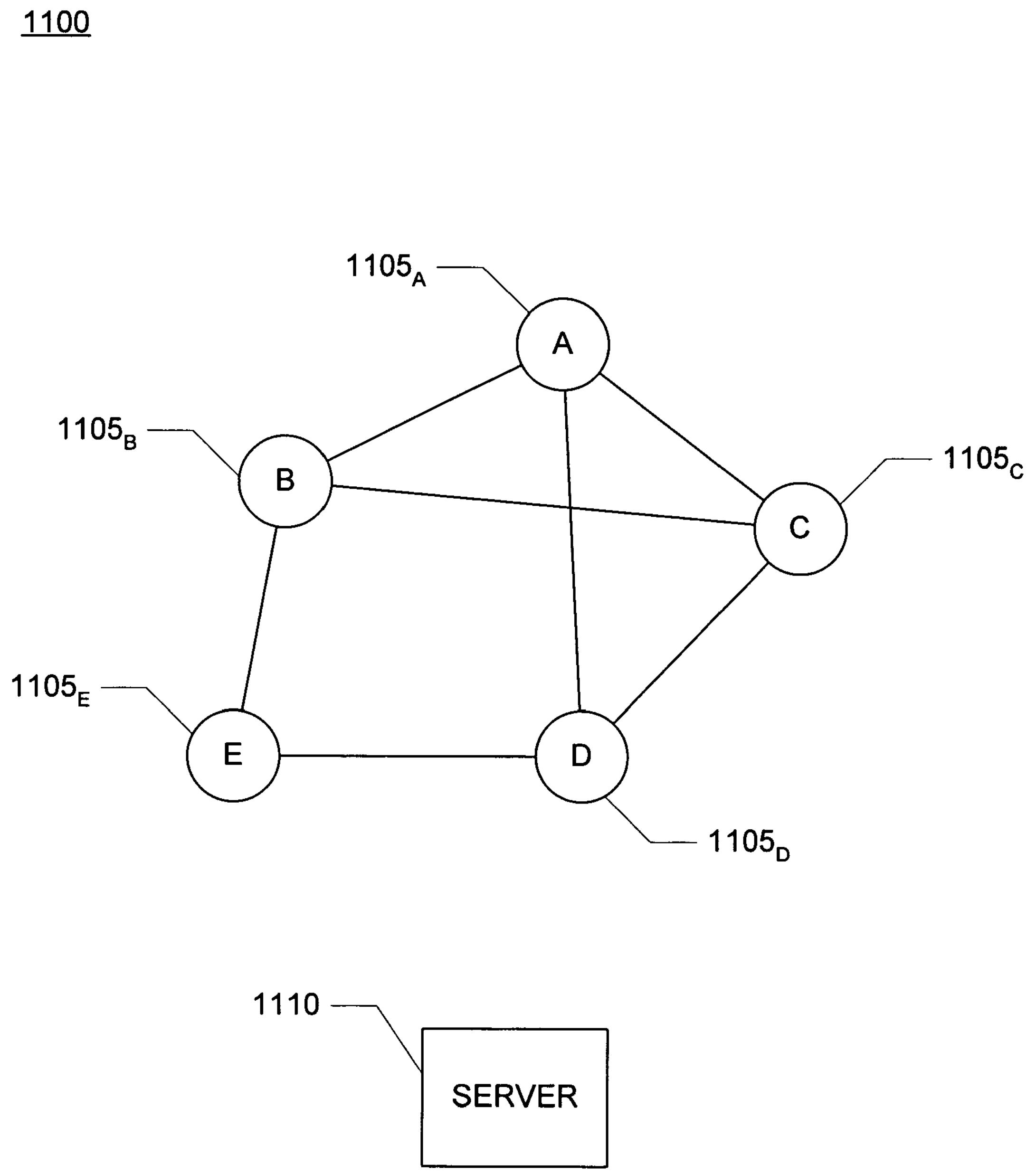


FIG. 15

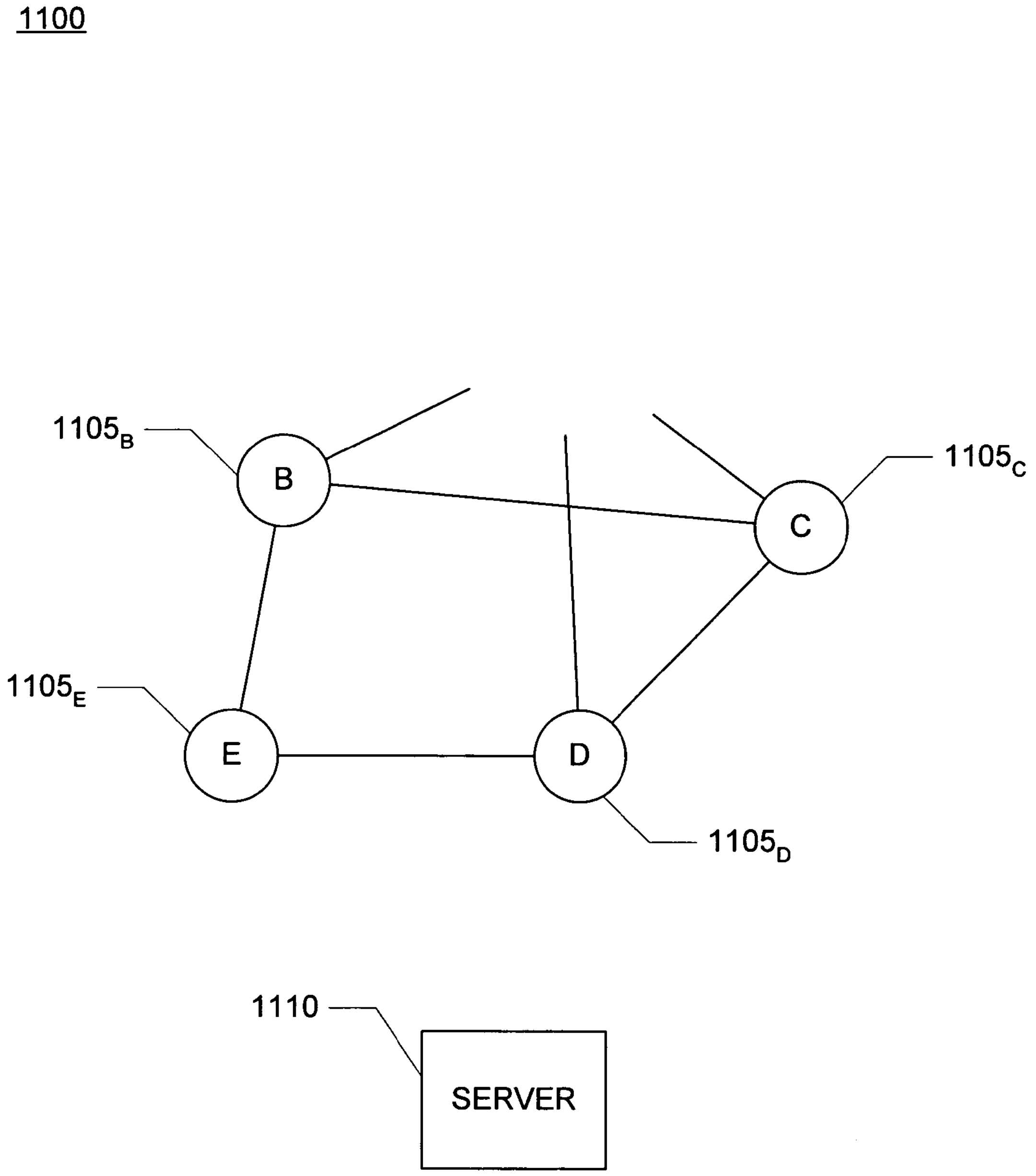


FIG. 16

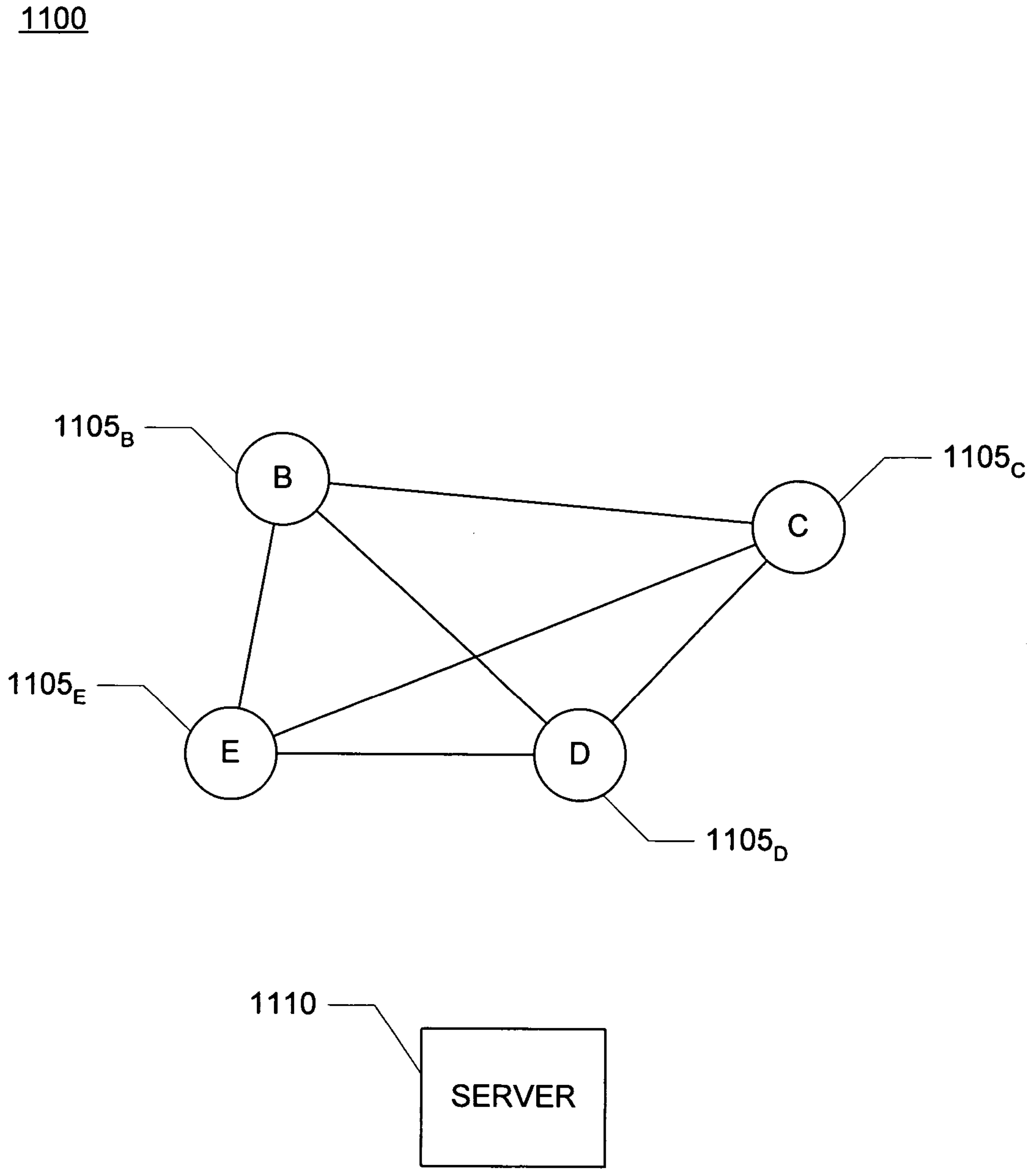


FIG. 17

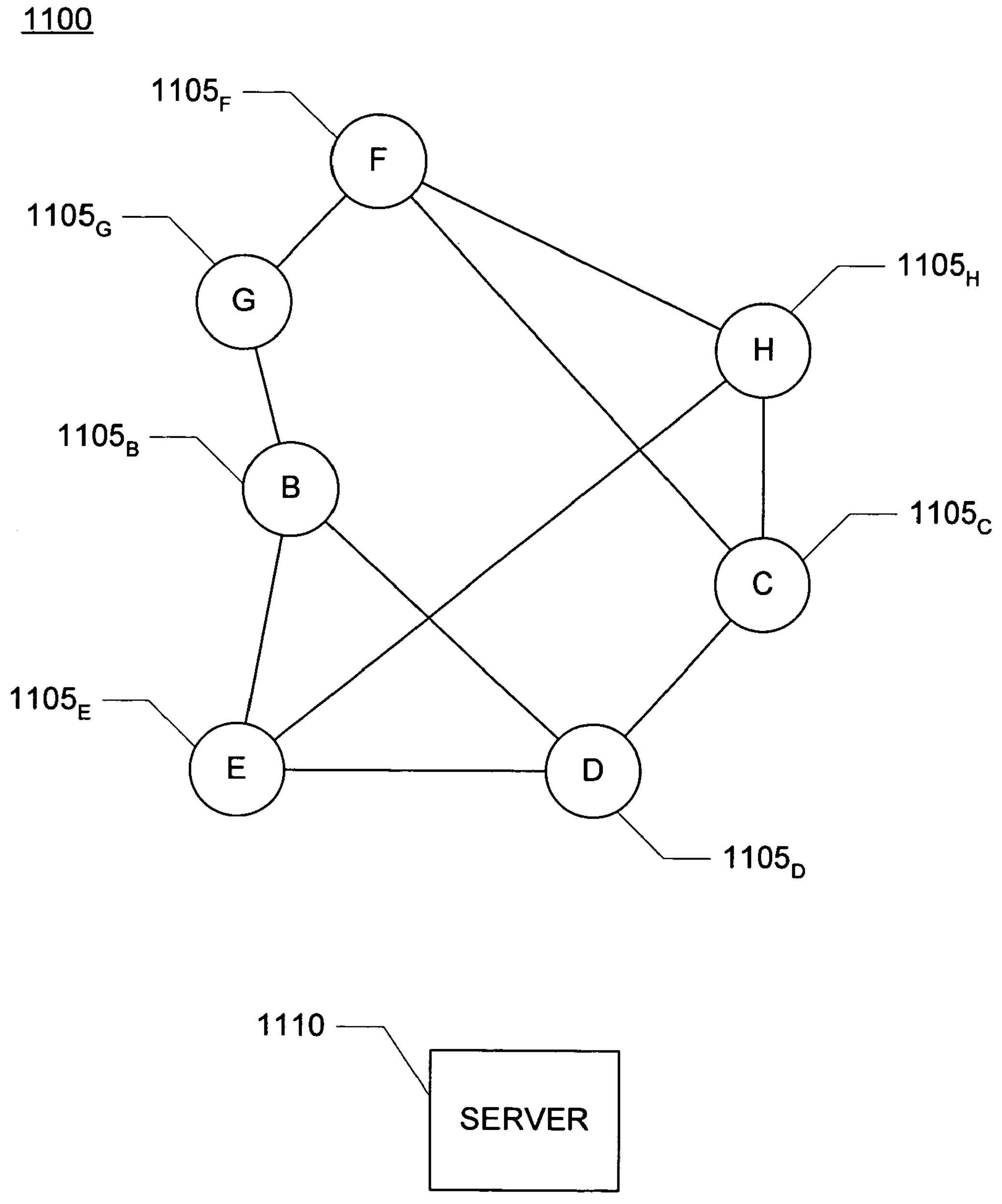

1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 18

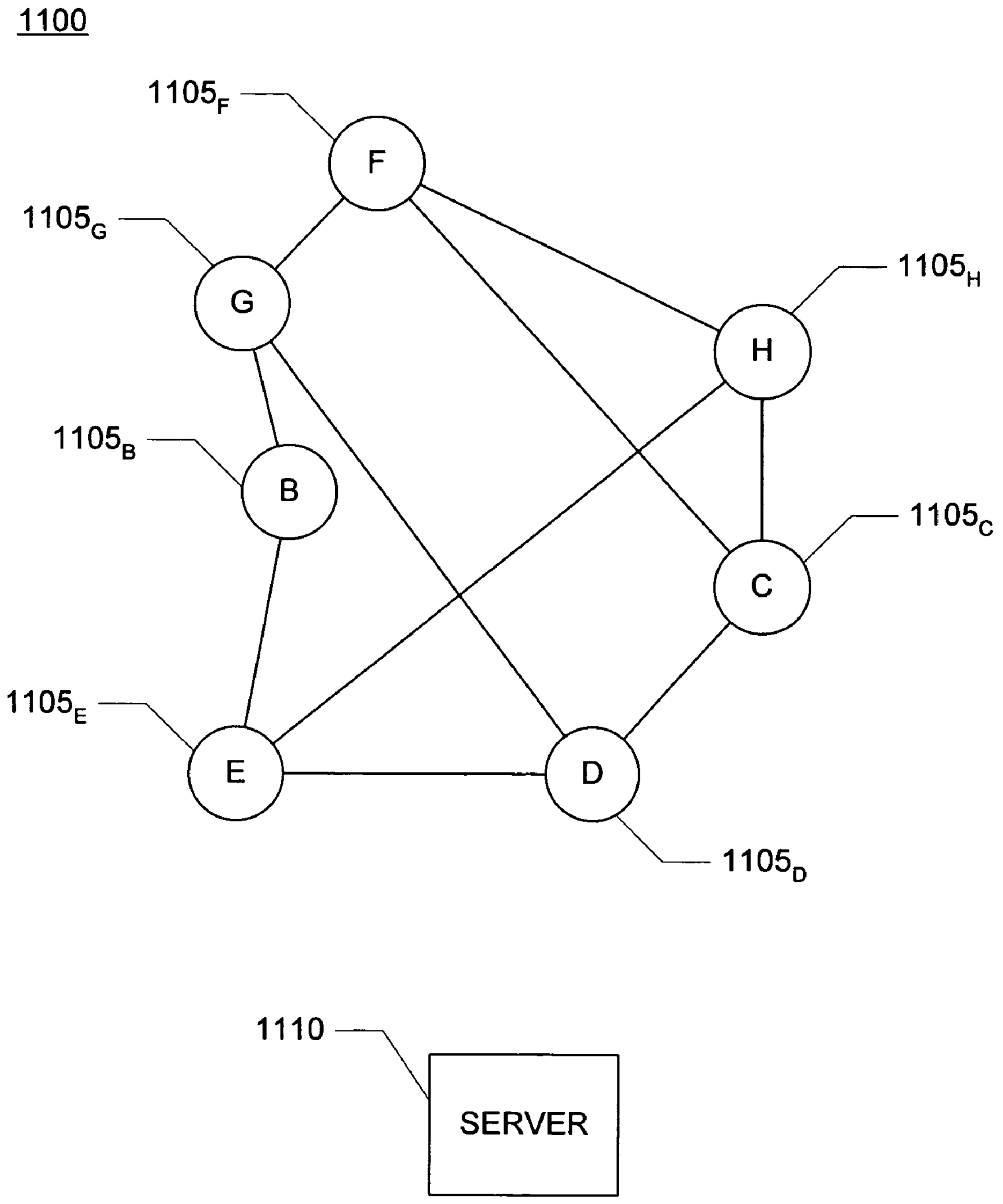

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 19

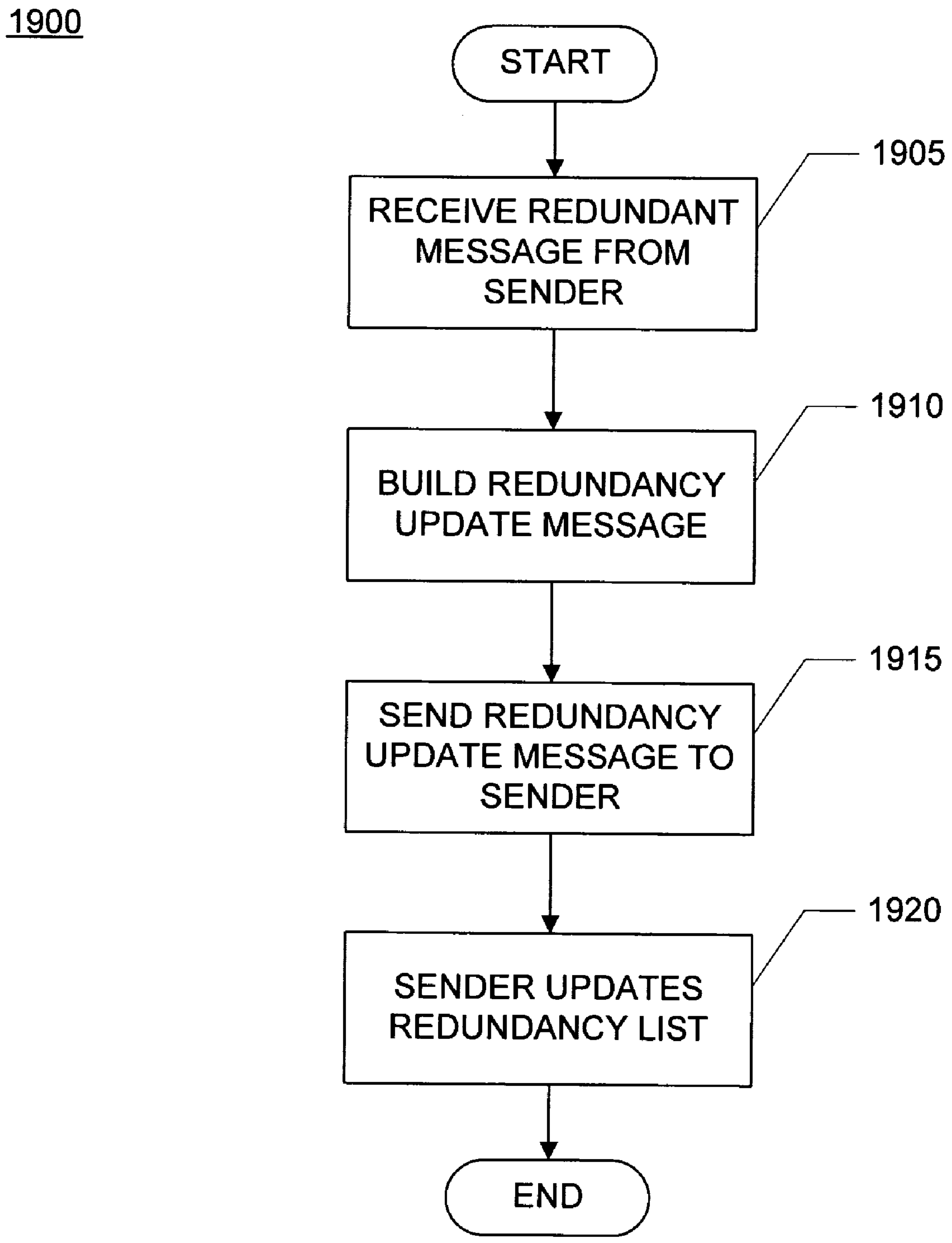

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

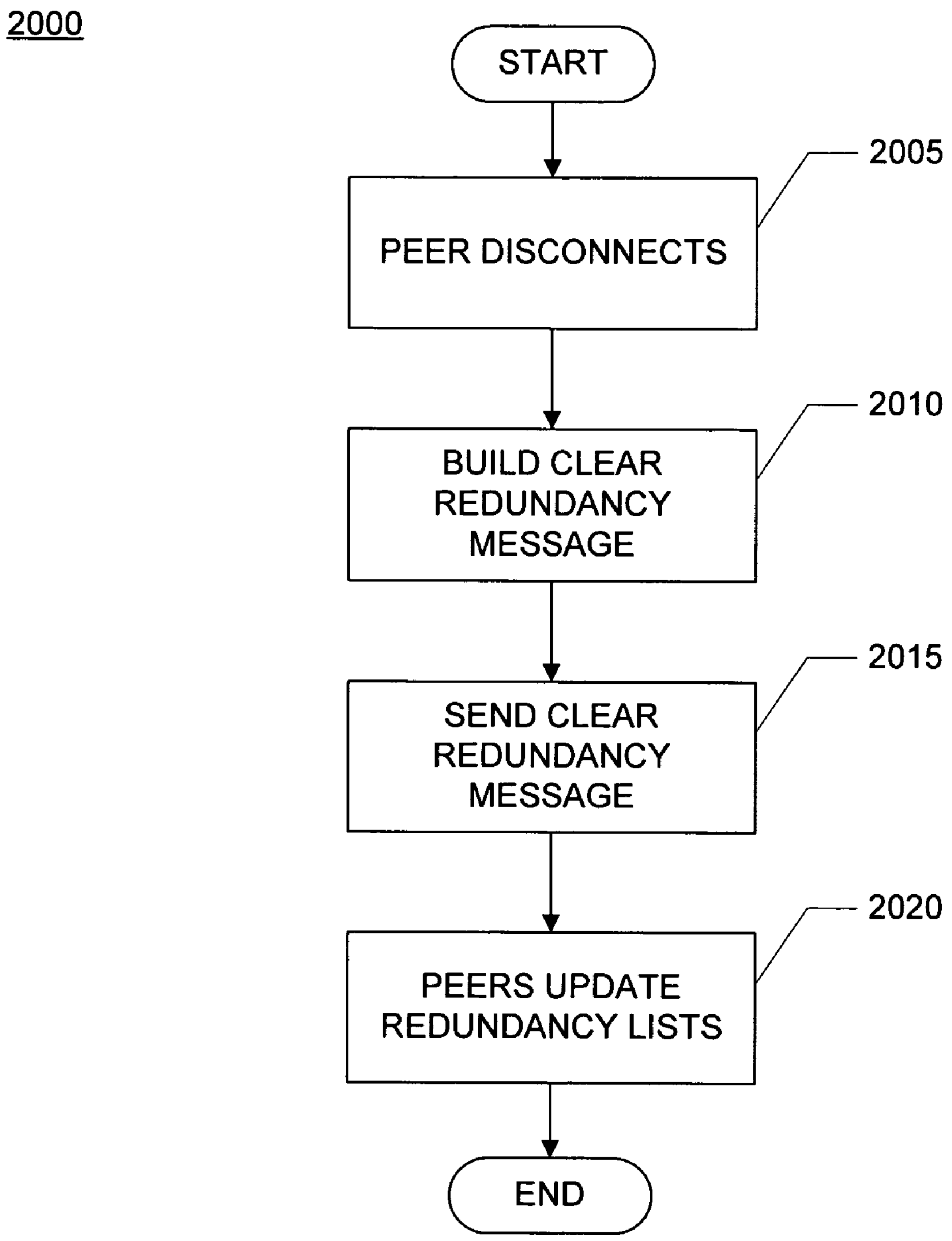

2000
START
PEER DISCONNECTS
2005
BUILD CLEAR REDUNDANCY MESSAGE
2010
SEND CLEAR REDUNDANCY MESSAGE
2015
PEERS UPDATE REDUNDANCY LISTS
2020
END

FIG. 21

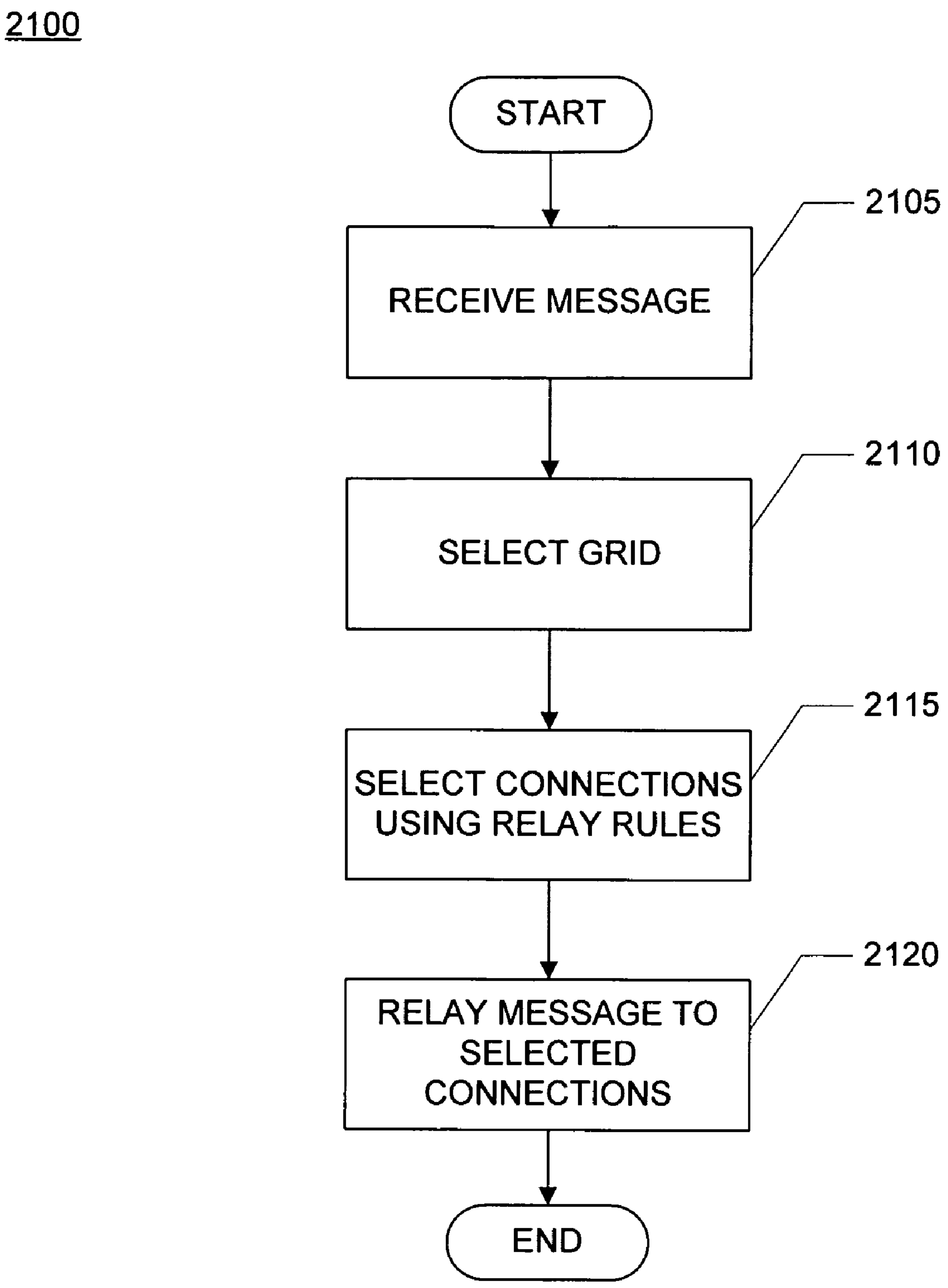

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

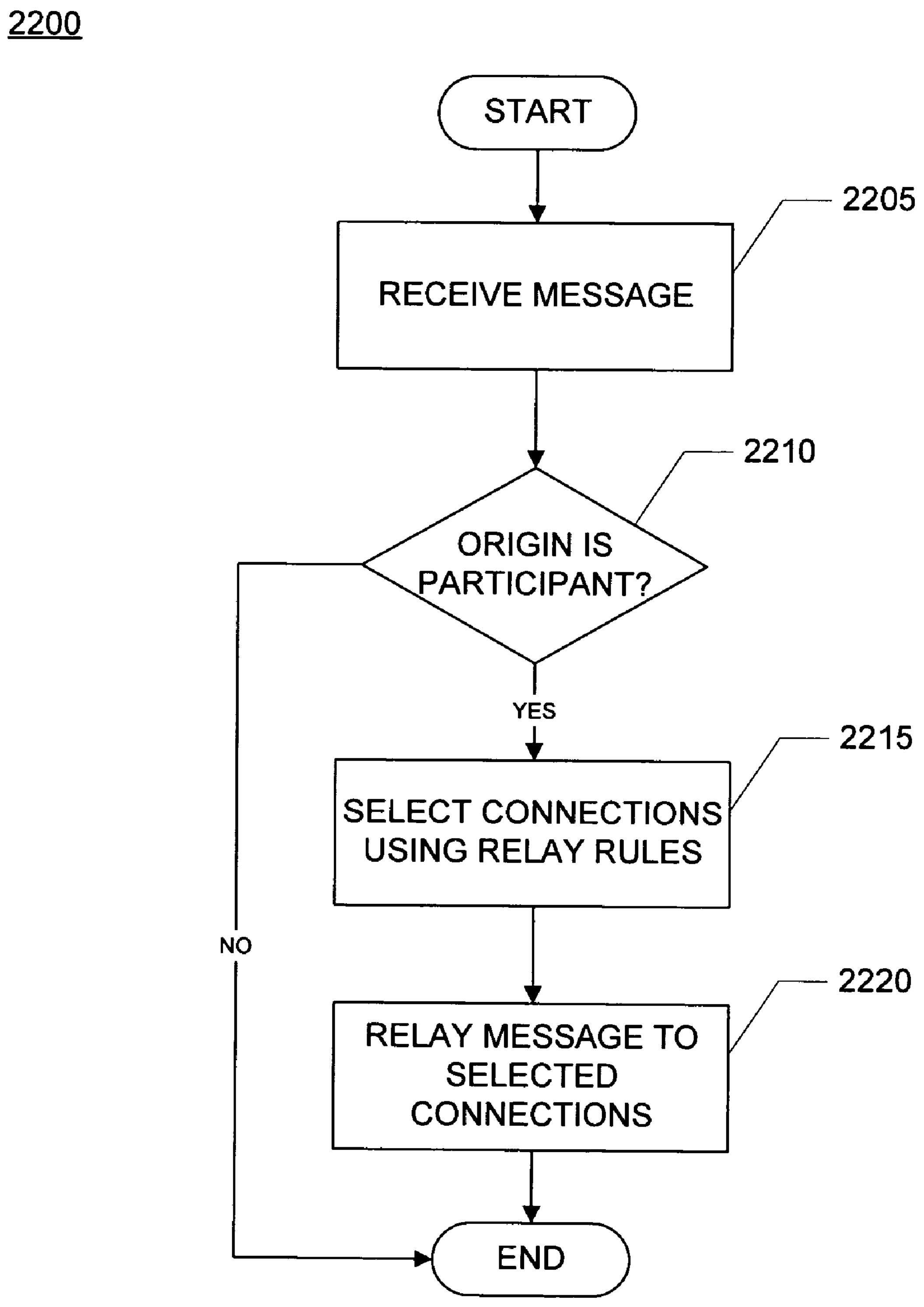

2200
START
RECEIVE MESSAGE
2205
ORIGIN IS PARTICIPANT?
2210
YES
NO
SELECT CONNECTIONS USING RELAY RULES
2215
RELAY MESSAGE TO SELECTED CONNECTIONS
2220
END

FIG. 23

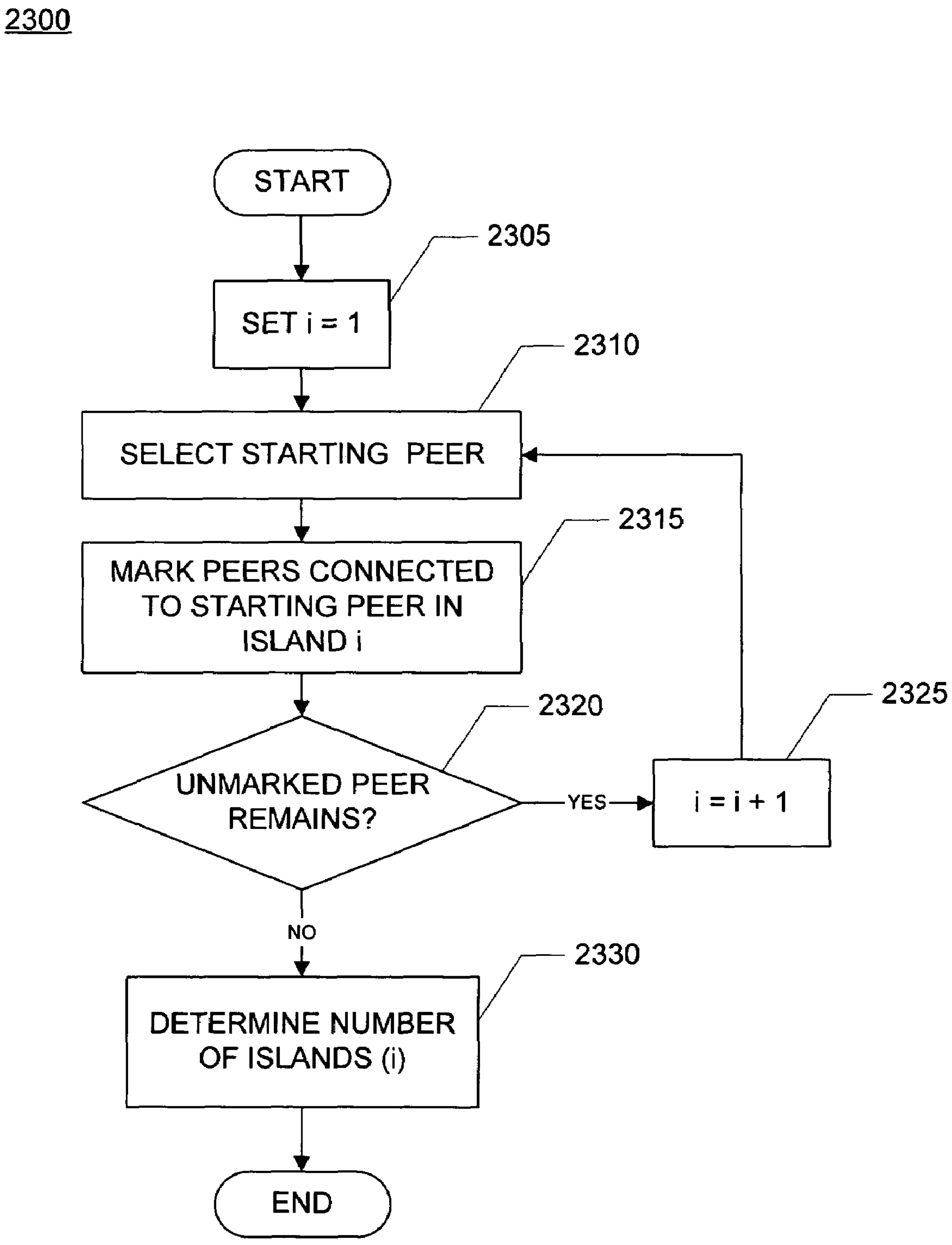

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

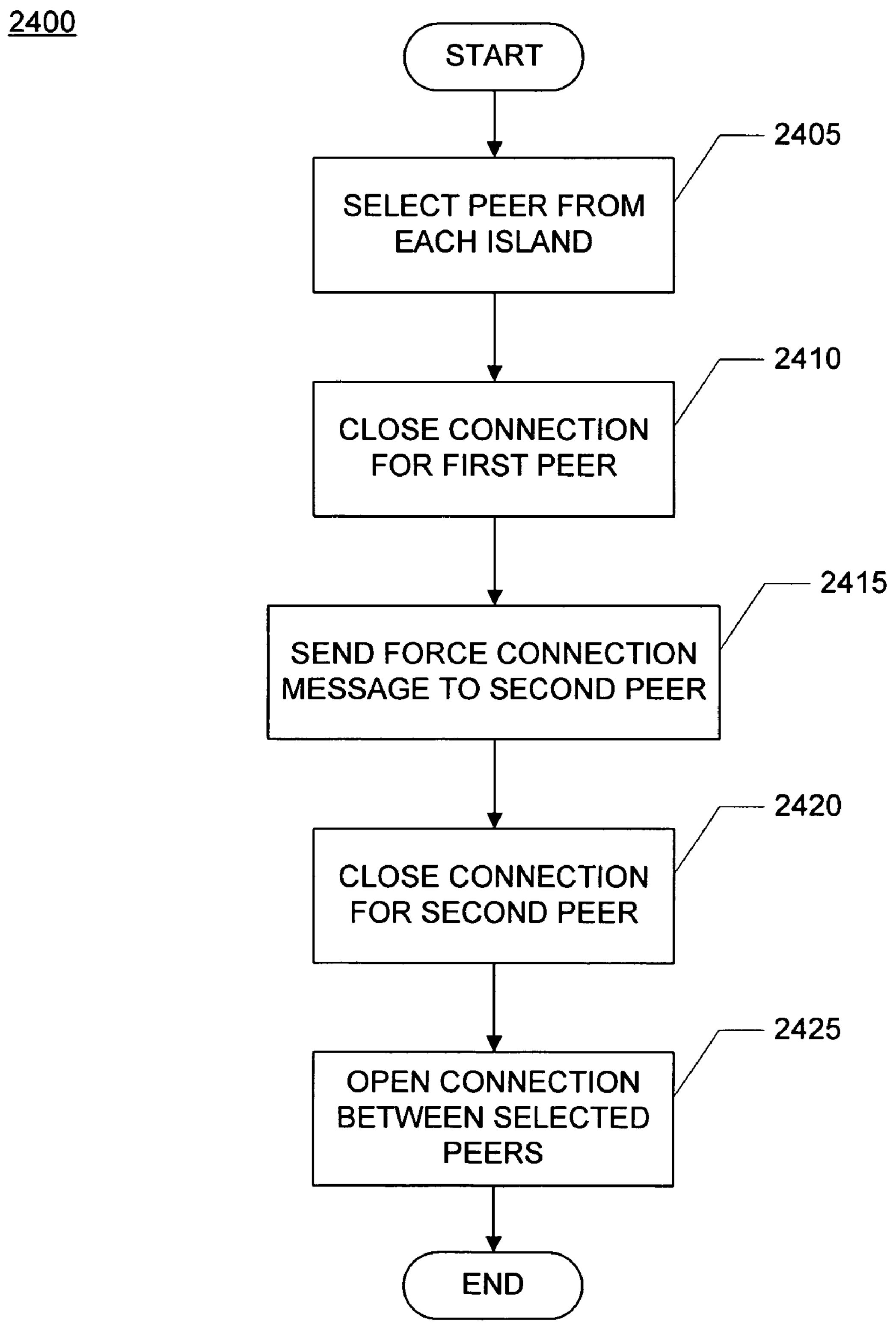

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

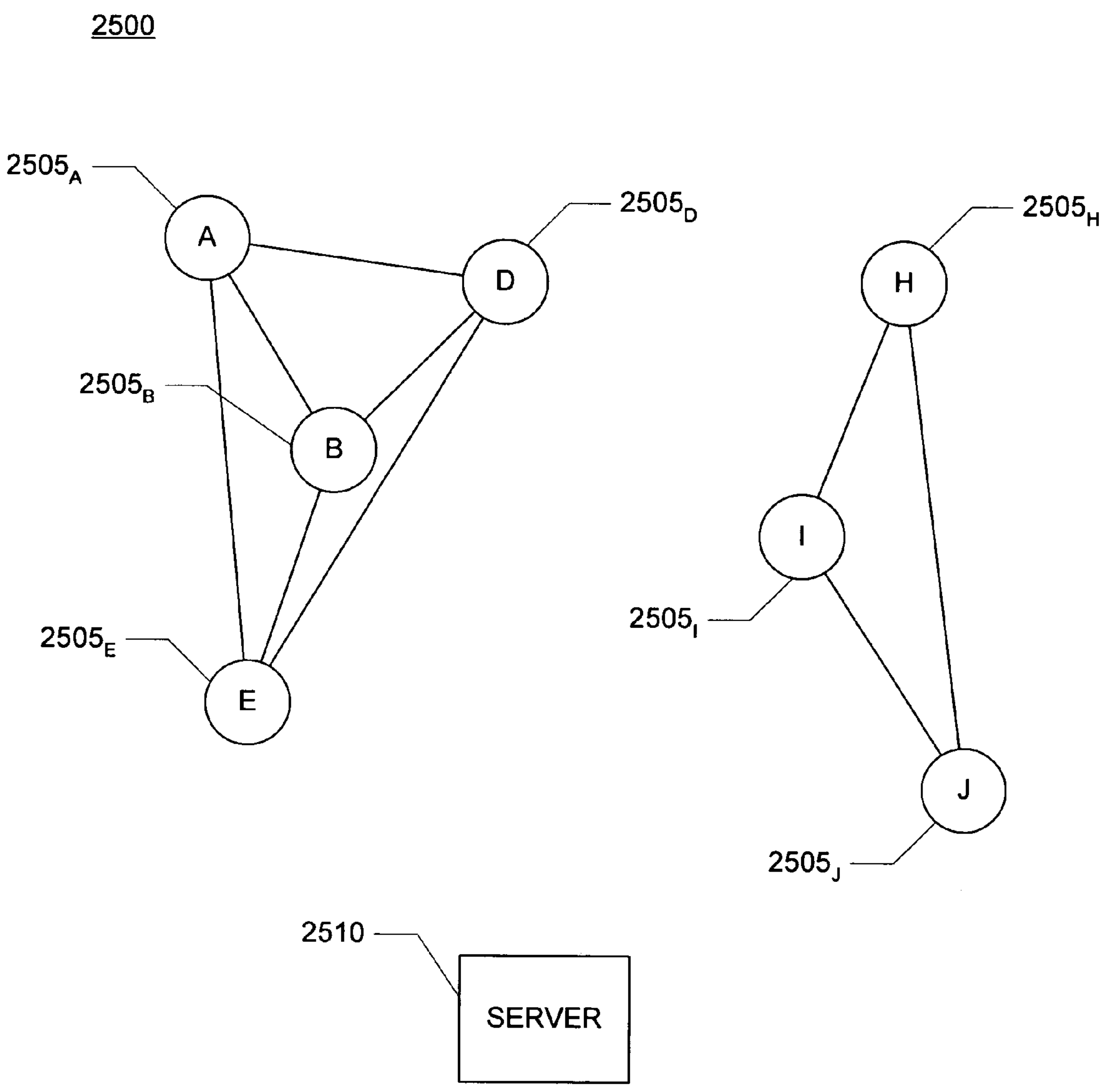

2500
2505$_A$
A
2505$_D$
D
2505$_B$
B
2505$_E$
E
2505$_H$
H
I
2505$_I$
J
2505$_J$
2510
SERVER

FIG. 26

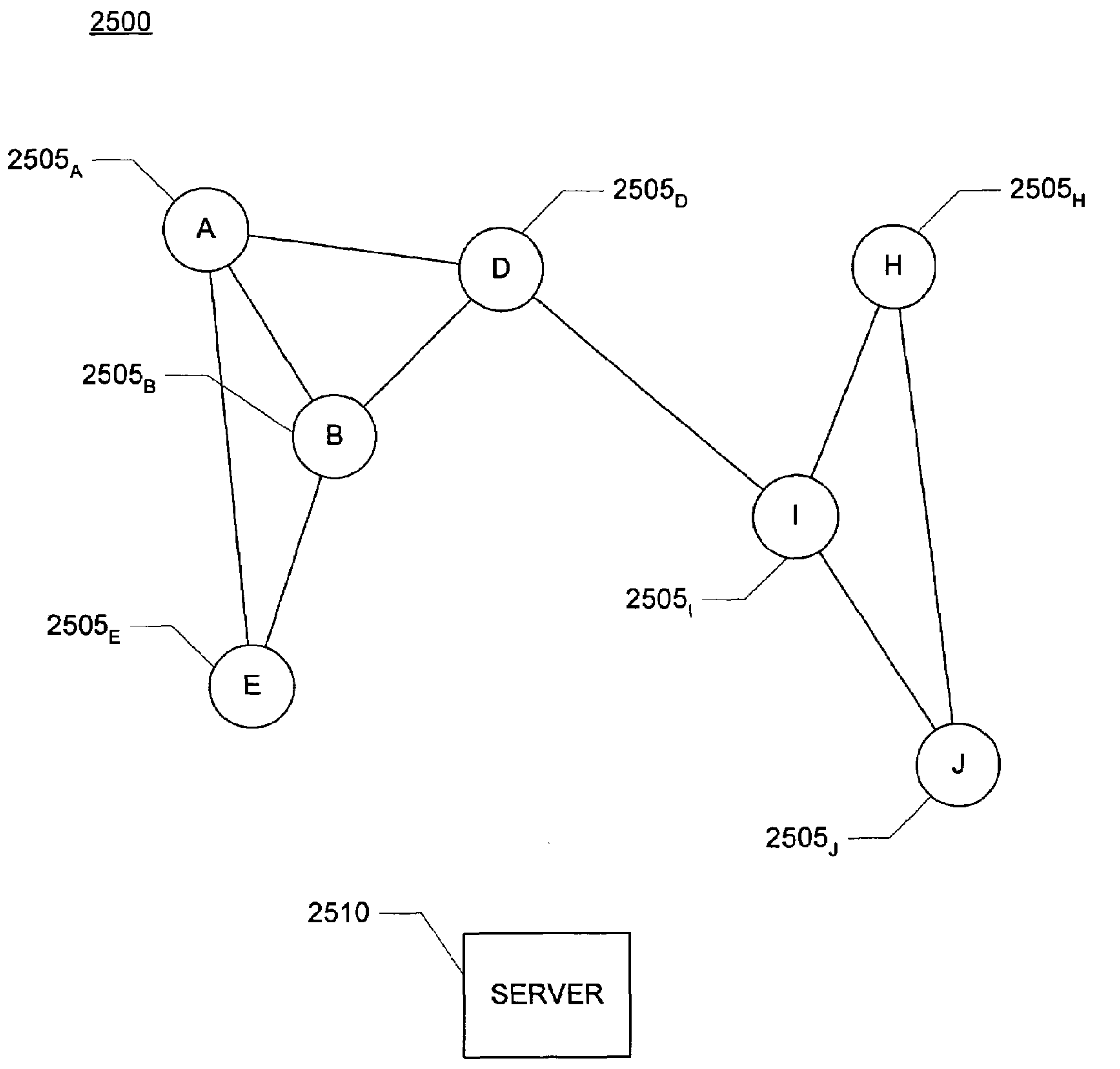

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 27

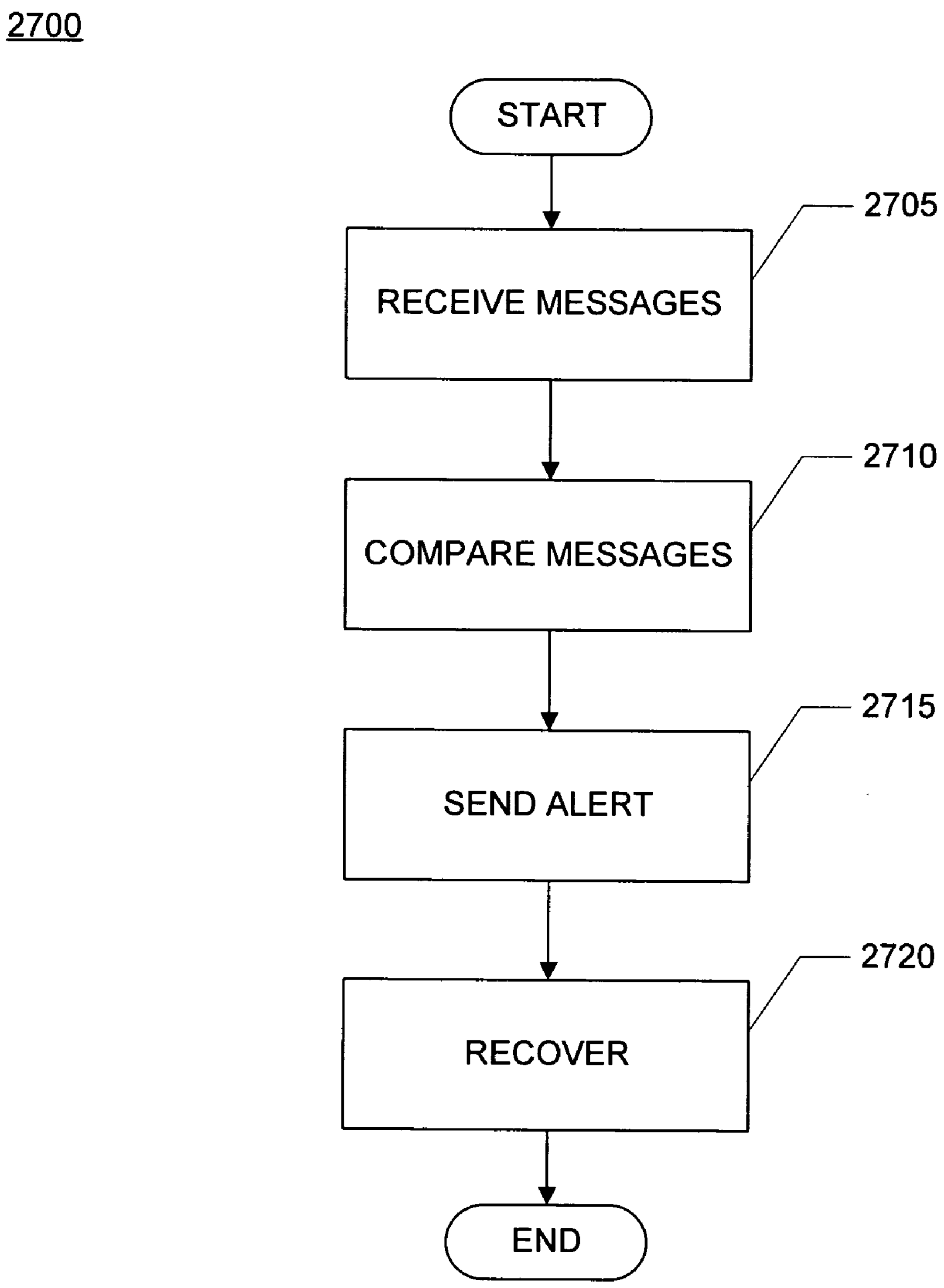

2700
START
2705
RECEIVE MESSAGES
2710
COMPARE MESSAGES
2715
SEND ALERT
2720
RECOVER
END

FIG. 28

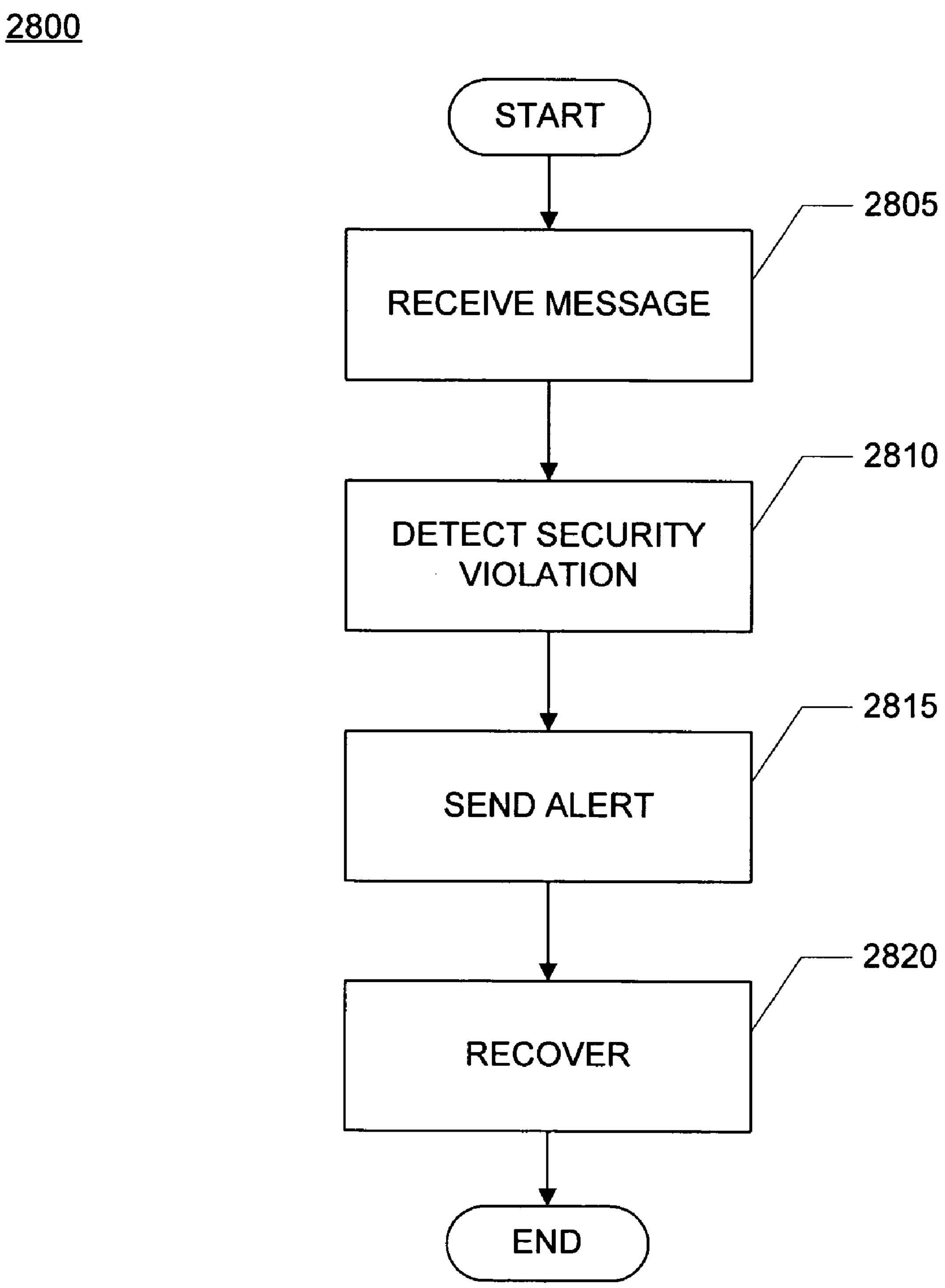

2800
START
2805
RECEIVE MESSAGE
2810
DETECT SECURITY VIOLATION
2815
SEND ALERT
2820
RECOVER
END

FIG. 29

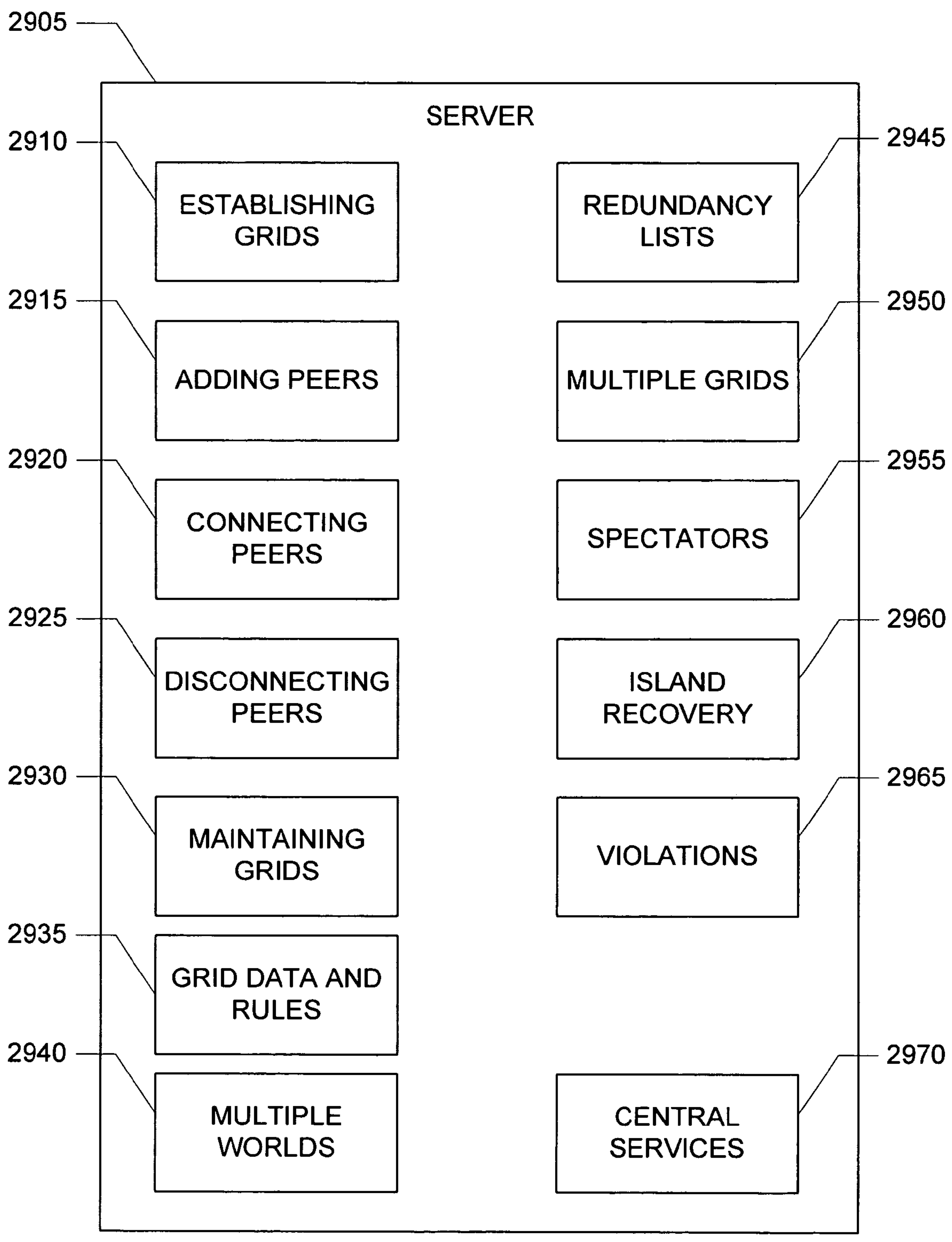

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

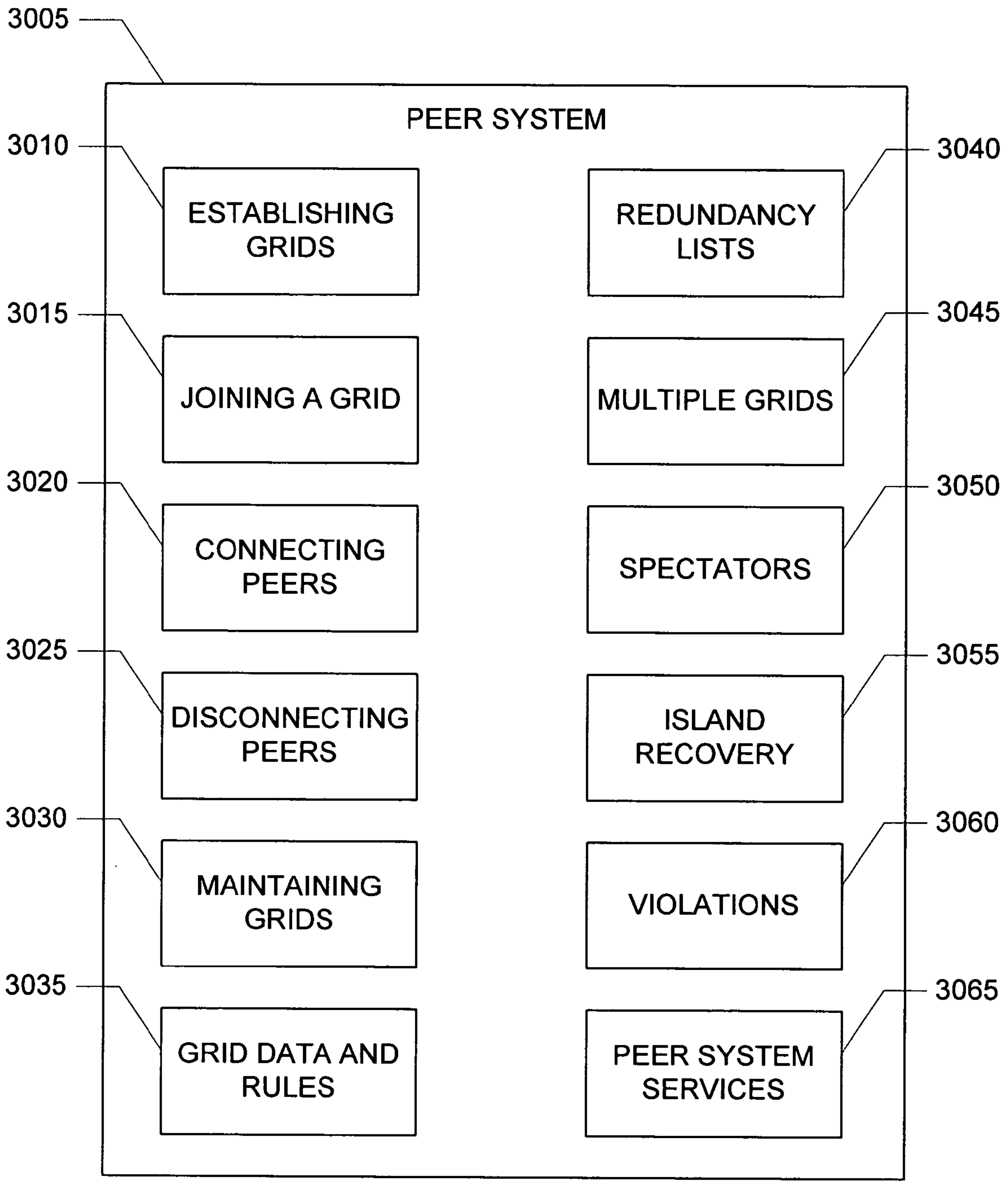

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

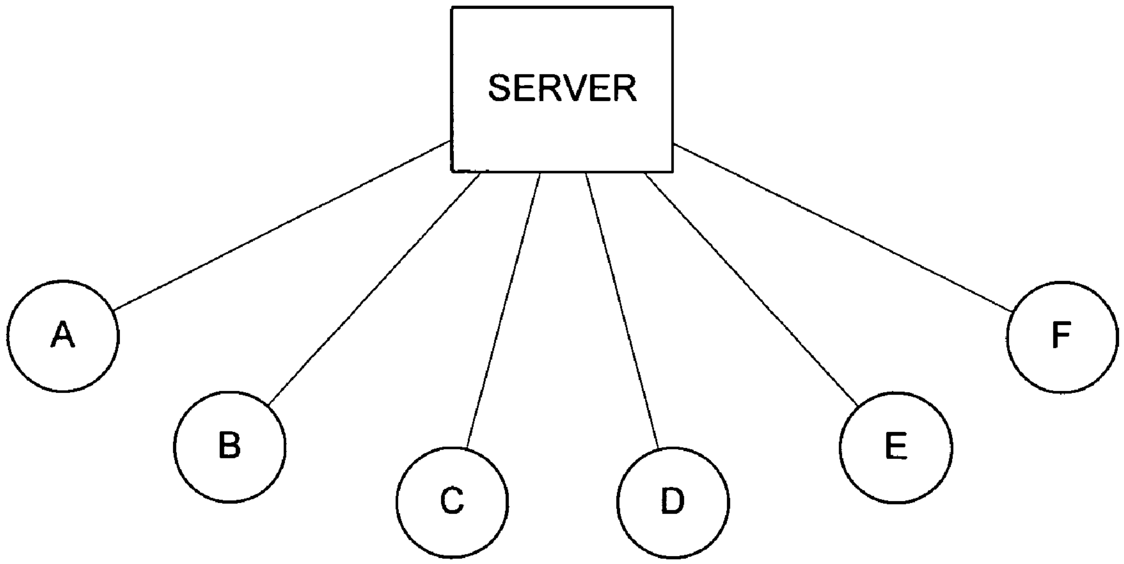
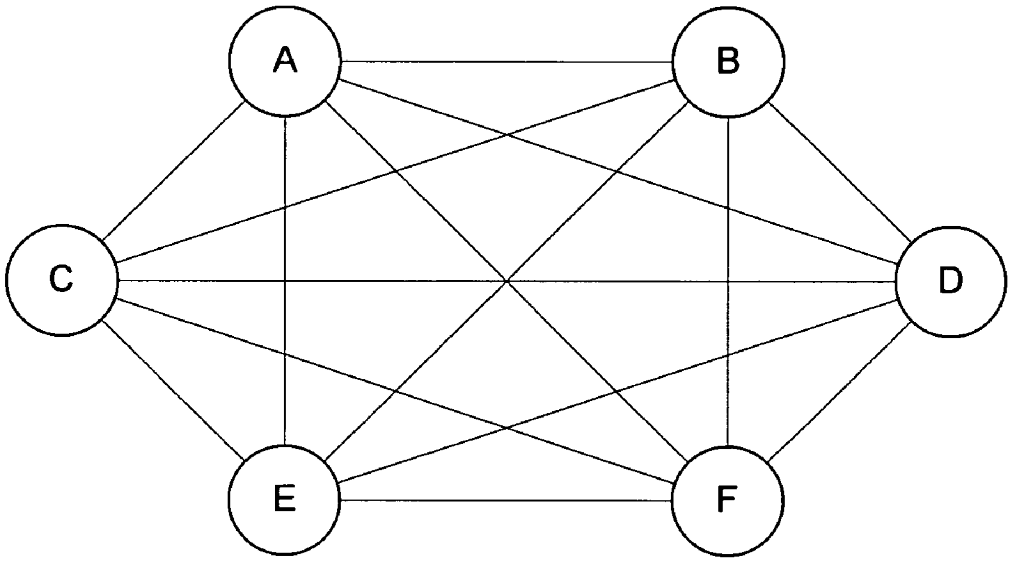


FIG. 31B



CONNECTING A PEER IN A PEER-TO-PEER RELAY NETWORK

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

This application is related to the U.S. applications Ser. No. 10/701,302, filed on Nov. 3, 2003, Ser. No. 10/701,014, filed on Nov. 3, 2003, Ser. No. 10/700,777, filed on Nov. 3, 2003, Ser. No. 10/701,298, filed on Nov. 3, 2003, and Ser. No. 10/700,797.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N−1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, and each peer system is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules.

In another implementation, a server for a peer-to-peer relay network includes: means for establishing a peer-to-peer relay network; means for adding a peer system to a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for tracking connections in a peer-to-peer relay network.

In another implementation, a peer system for a peer-to-peer relay network includes: means for relaying data to any other peer systems connected to said peer system in a peer-to-peer relay network; means for establishing a peer-to-peer relay

network; means for joining a peer-to-peer relay network; means for connecting to another peer system in a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for disconnecting from another peer system connected to said peer system in a peer-to-peer relay network.

In another implementation, a method of relaying data in a peer-to-peer relay network includes: receiving data at a relaying peer system from a sending peer system connected to said relaying peer system in a peer-to-peer relay network; applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and relaying said data to any peer systems selected by applying said set of one or more relay rules.

In another implementation, a method of adding a peer system to a peer-to-peer relay network includes: opening a connection between a server and a joining peer system; providing grid information to said joining peer system indicating one or more established peer-to-peer relay networks; receiving a grid selection from said joining peer system indicating a selected peer-to-peer relay network, wherein said selected peer-to-peer relay network has one or more member peer systems; providing network addresses of each of said one or more member peer systems to said joining peer system; and receiving a connection update from said joining peer system indicating to which member peer systems said joining peer system is connected; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to a connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of joining a peer-to-peer relay network includes: sending a join message from a joining peer system to each of one or more member peer systems in a peer-to-peer relay network; receiving a join response from at least one of said one or more member peer systems, wherein each join response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting one or more member peer systems up to a connection limit according to a set of one or more connection rules; opening a connection with each selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of establishing a peer-to-peer relay network includes: opening a connection between said server and an establishing peer system, wherein the establishing peer system is one of said member peer systems; sending a request to create said peer-to-peer relay network from said establishing peer system to said server; receiving a creation confirmation at said establishing peer system from said server; wherein said establishing peer system stores a connection limit defining a number of other peer systems up to which said establishing peer system is permitted to connect, and said establishing peer system stores a set of one or more relay rules for relaying data to other peer systems connected to said establishing peer system.

In another implementation, a method of connecting peer systems in a peer-to-peer relay network includes: sending a connection available message from a disconnected peer system to one or more member peer systems in a peer-to-peer relay network when said disconnected peer system has a number of open connections to member systems that is less than a connection limit; receiving a connection available response from at least one of said one or more member peer systems, wherein each connection available response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting a member peer system according to a set of one or more connection rules; opening a connection with said selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of maintaining a peer-to-peer relay network includes: sending a maintenance message from a peer system to each of one or more connected peer systems connected to said peer system in a peer-to-peer relay network; evaluating any responses received from said one or more connected peer systems; and closing the connection between said peer system and a connected peer system if the response from that connected peer system is not acceptable; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit and each peer system stores a set of one or more relay rules for relaying data to the other peer systems connected to that peer system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent applications Ser. No. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed 31 Jul. 2002, and Ser. No. 10/359,359 ("Multi-User Application Programming Interface"), filed 4 Feb. 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system $\mathbf{105}_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F. depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and

compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends ajoin message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. Ajoin response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of **3**, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new

peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet).

In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105_A** (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105_B** (peer B) has joined the grid **1100**. When peer B joins,

peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B.

Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method of adding a peer system to a peer-to-peer relay network, comprising:

opening a connection between a server and a joining peer system;

providing grid information to said joining peer system indicating one or more established peer-to-peer relay networks;

receiving a grid selection from said joining peer system indicating a selected peer-to-peer relay network, wherein said selected peer-to-peer relay network has one or more member peer systems;

providing network addresses of each of said one or more member peer systems to said joining peer system; and

receiving a connection update from said joining peer system indicating to which member peer systems said joining peer system is connected;

wherein each member peer system is connected to a number of other member peer systems that is less than or equal to a connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system, and

wherein the joining peer system sends a join request to the one or more member peer systems and receives either a positive or a negative responses from the member peer system,

selecting a member peer system from which a negative join response has been received as a force connection peer system;

sending a force connection request to said force connection peer system, wherein said force connection request requests that said force connection peer system close one of the open connection of said force connection peer system,

wherein selecting said force connection peer system includes applying said set of one or more connection rules to the member peer systems that sent negative join responses.

2. The method of claim **1**, further comprising:

opening a connection between said server and an establishing peer system, wherein the establishing peer system is one of said member peer systems;

receiving a request to create said peer-to-peer relay network from said establishing peer system;

registering said peer-to-peer relay network in storage; and

sending a creation confirmation to said establishing peer system.

3. A method of joining a peer-to-peer relay network, comprising:

sending a join message from a joining peer system to each of one or more member peer systems in a peer-to-peer relay network;

receiving a join response from at least one of said one or more member peer systems, wherein each join response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection;

selecting one or more member peer systems up to a connection limit according to a set of one or more connection rules;

opening a connection with each selected member peer system;

wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system, and

wherein selecting one or more member peer systems includes;

selecting a member peer system from which a negative join response has been received as a force connection peer system;

sending a force connection request to said force connection peer system, wherein said force connection request requests that said force connection peer system close one of the open connections of said force connection peer system,

wherein selecting said force connection peer system includes applying said set of one or more connection rules to the member systems that sent negative join responses.

4. The method of claim **3** wherein:

a member peer system has an available connection if the member peer system has a number of open connections to other member peer systems that is less than said connection limit.

5. The method of claim **3**, wherein:

selecting one or more member peer systems includes storing a response time for each received join response.

6. The method of claim **3**, wherein:

selecting one or more member peer systems includes:

selecting the member peer system from which the positive join response that is received first by said joining peer system, and selecting the member peer system from which the positive join response that is received last by said joining peer system within a time limit.

7. The method of claim **6**, wherein:

selecting one or more member peer systems includes substantially randomly selecting additional member peer systems up to said connection limit from among the remaining unselected member peer systems from which positive joint response have been received.

8. The method of claim **6**, wherein:

selecting one or more member peer systems includes selecting additional member peer systems up to said connection limit from among the remaining unselected member peer systems from which positive joint response have been received in the order in which the positive joint responses were received.

9. The method of claim **3**, comprising,

receiving a force connection confirmation from said force connection peer system.

10. The method of claim **3**, further comprising:

opening a connection between a server and said joining peer system;

receiving grid information at said joining peer system indicating one or more established peer-to-peer relay networks;

sending a grid selection from said joining peer system to said server indicating a selected peer-to-peer relay network, wherein said selected peer-to-peer relay network has one or more member peer systems;

receiving network addresses of each of said one or more member peer systems at said joining peer system; and

sending a connection update from said joining peer system indicating to which member peer systems said joining peer system is connected.

11. A method of establishing a peer-to-peer relay network, comprising:

opening a connection between a server and an establishing peer system, wherein the establishing peer system is one of said member peer systems;

sending a request to create said peer-to-peer relay network from said establishing peer system to said server;

receiving a creation confirmation at said establishing peer system from said server;

wherein said establishing peer system stores a connection limit defining a number of other peer systems up to which said establishing peer system is permitted to connect, and said establishing peer system stores a set of one or more relay rules for relaying data to other peer systems connected to said establishing peer system, and

wherein the joining peer system sends a join request to the one or more member peer systems and receives either a positive or a negative response from the member peer system,

selecting a member peer system from which a negative join response has been received as a force connection peer system;

sending a force connection request to said force connection peer system, wherein said force connection request requests that said force connection peer system close one of the open connections of said force connection peer system,

wherein selecting said force connection peer system includes applying said set of one or more connection rules to the member peer systems that sent negative join responses.

12. A method of connecting peer systems in a peer-to-peer relay network comprising:

sending a connection available message from a disconnected peer system to one or more member peer systems in a peer-to-peer relay network when said disconnected peer system has a number of open connections to member systems that is less than a connection limit;

receiving a connection available response from at least one of said one or more member peer systems, wherein each connection available response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection;

selecting a member peer system according to a set of one or more connection rules;

opening a connection with said selected member peer system;

wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system, and

wherein selecting one or more member peer systems includes:

selecting a member peer system from which a negative join responses has been received as a force connection peer system;

sending a force connection request to said force connection peer system, wherein said force connection request requests that said force connection peer system close one of the open connections of said force connection peer system,

wherein selecting said force connection peer system includes applying said set of one or more connections rules to the member peer systems that sent negative join responses.

13. The method of claim **12**, further comprising:

closing a connection by said disconnected peer system.

14. The method of claim **12**, wherein:

a member peer system has an available connection if the member peer system has a number of open connections to other member peer systems that is less than said connection limit.

15. The method of claim **12**, wherein:

selecting a member peer system includes storing a response time for each received connection available response.

16. The method of claim **12**, wherein:

selecting a member peer systems includes selecting the member peer system from which the positive connection available response that is received first by said disconnected peer system.

17. The method of claim **12**, wherein:

selecting a member peer systems includes not selecting a member peer system from which said disconnected peer system has disconnected within a disconnection time period.

18. The method of claim **12**, comprising,

receiving a force connection confirmation from said force connection peer system.

19. The method of claim **12**, further comprising:

sending an update to a server indicating a connection has been opened between said disconnected peer system and said selected member peer system.

* * * * *

EXHIBIT 88

US007725599B2

(12) **United States Patent**
Van Datta

(10) **Patent No.:** **US 7,725,599 B2**
(45) **Date of Patent:** **May 25, 2010**

(54) **PEER-TO-PEER DATA RELAY**

(75) Inventor: **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America, Inc.**, Foster City, 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.: **12/120,397**

(22) Filed: **May 14, 2008**

(65) **Prior Publication Data**

US 2008/0222250 A1 Sep. 11, 2008

Related U.S. Application Data

(60) Division of application No. 10/970,120, filed on Oct. 20, 2004, which is a continuation-in-part of application No. 10/700,798, filed on Nov. 3, 2003.

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/243**; 709/200

(58) **Field of Classification Search** 709/200, 709/238, 243, 244

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,667,972	B1 *	12/2003	Foltan et al.	370/354
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1	12/2002	Traversat et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999

(Continued)

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

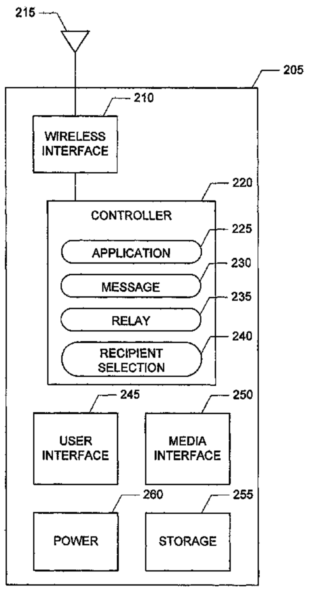
(Continued)

Primary Examiner—Salad Abdullahi
Assistant Examiner—El Hadji M Sall
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for relaying data in a peer-to-peer network. In one implementation, a wireless device includes: an antenna; a wireless communication interface connected to said antenna and supporting wireless communication across a wireless connection provided by said antenna; storage supporting storing data; and a controller connected to said wireless interface and to said storage, supporting an application service, a message service, and a relay service for relay messages; wherein said application service provides execution and management of one or more application programs accessible by said controller using application data stored in said storage, said message service provides building messages and processing received messages, and said relay service provides building a new relay message indicating a selected recipient to which the wireless device does not have a direct wireless connection, sending a built new relay message, and sending a received relay message that indicates a recipient other than the wireless device.

17 Claims, 7 Drawing Sheets



U.S. PATENT DOCUMENTS

2003/0027634	A1 *	2/2003	Matthews, III	463/39
2003/0046292	A1 *	3/2003	Subramanian et al.	707/100
2003/0055892	A1	3/2003	Huitema et al.	
2003/0079003	A1	4/2003	Burr	
2003/0220981	A1 *	11/2003	Nakamura et al.	709/217
2003/0227939	A1 *	12/2003	Yukie et al.	370/465
2004/0207880	A1 *	10/2004	Thakur	358/3.05
2005/0026698	A1 *	2/2005	Pirich et al.	463/43

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" Globecom 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups" Proceedings of the 7TH ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. Conf. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

"User's Guide HP Ipaq Pocket Prestige Crystal H5100 and H5500, part nbr. 304926-001" 'Online! May 2003, pp. 6.1-6.4, XP002344472 Retrieved from the Internet:URL:www.hp.org>' retrieved on Sep. 12, 2005! p. 6.1-p. 6.4.

* cited by examiner

FIG. 1A

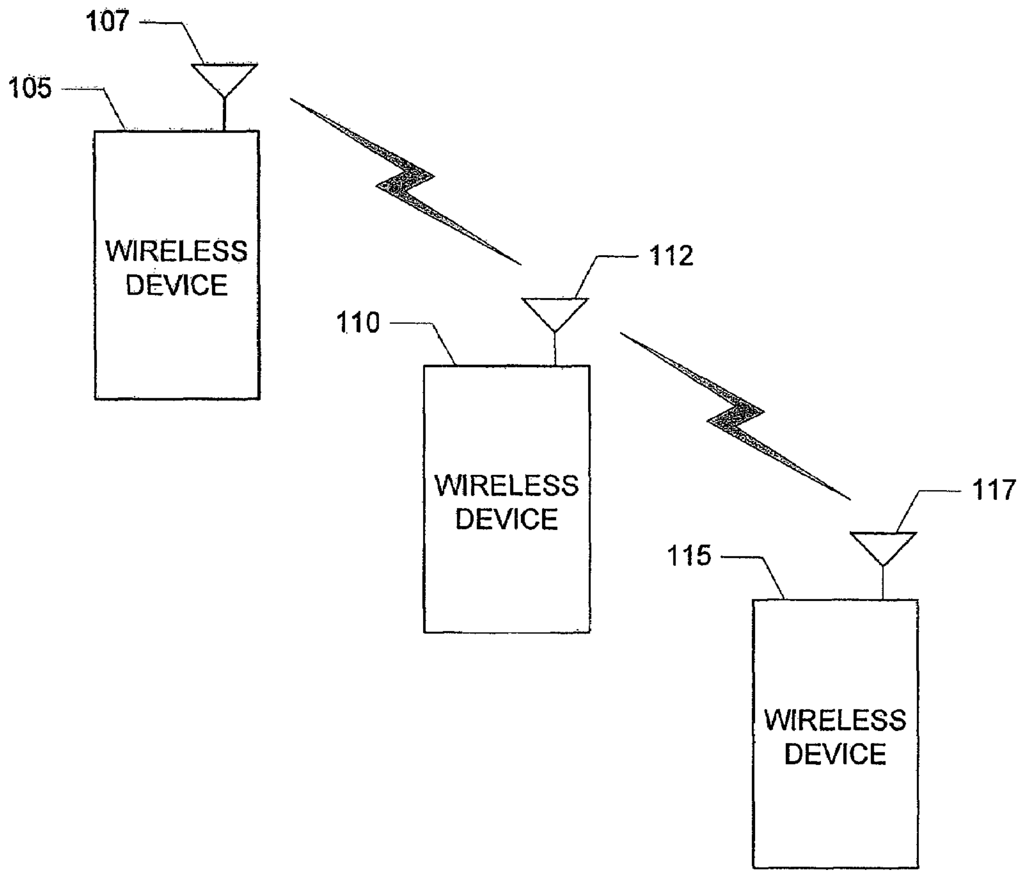

107
105
WIRELESS DEVICE
112
110
WIRELESS DEVICE
117
115
WIRELESS DEVICE

FIG. 1B

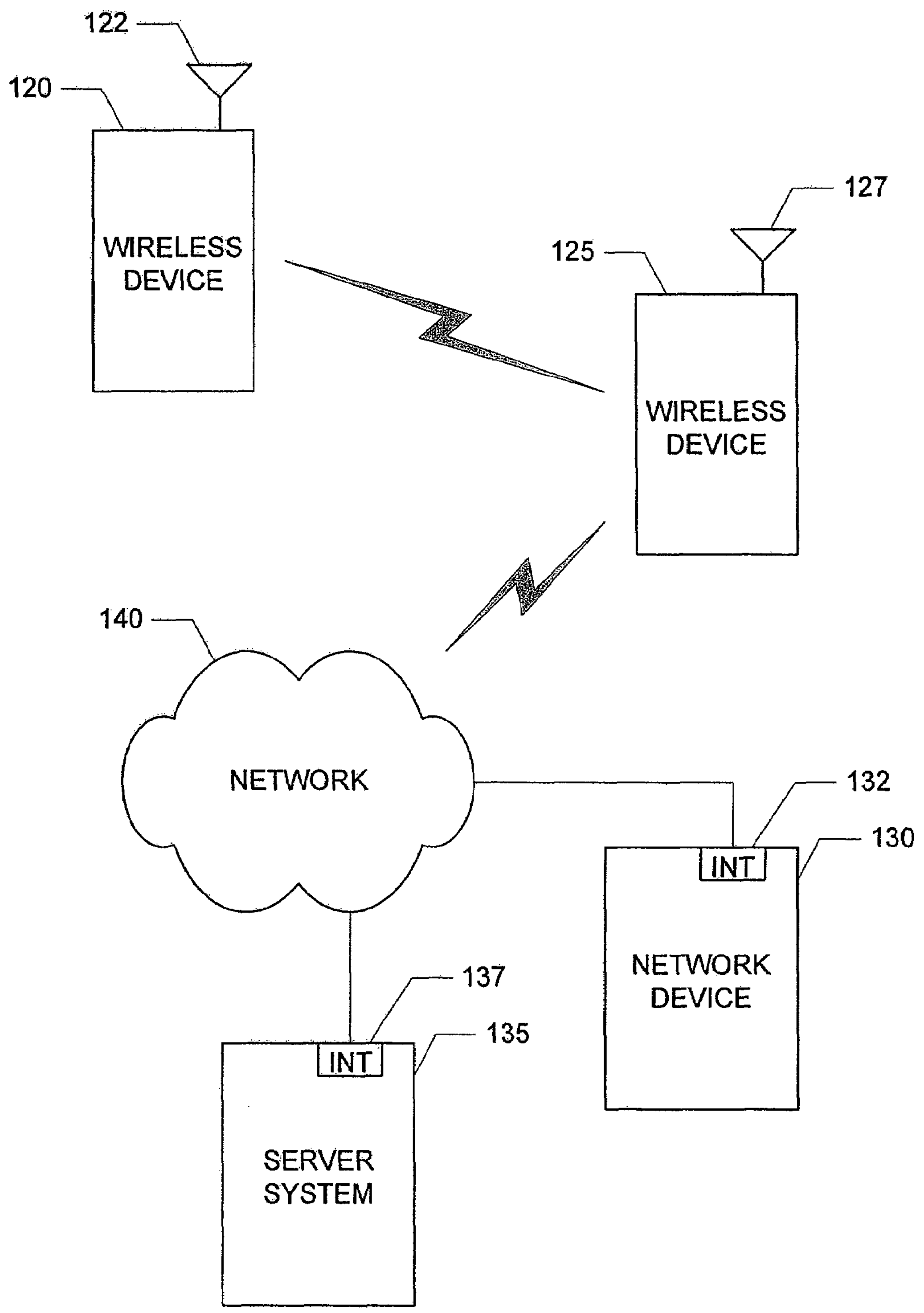

122
120
WIRELESS DEVICE
127
125
WIRELESS DEVICE
140
NETWORK
132
130
INT
NETWORK DEVICE
137
135
INT
SERVER SYSTEM

FIG. 2

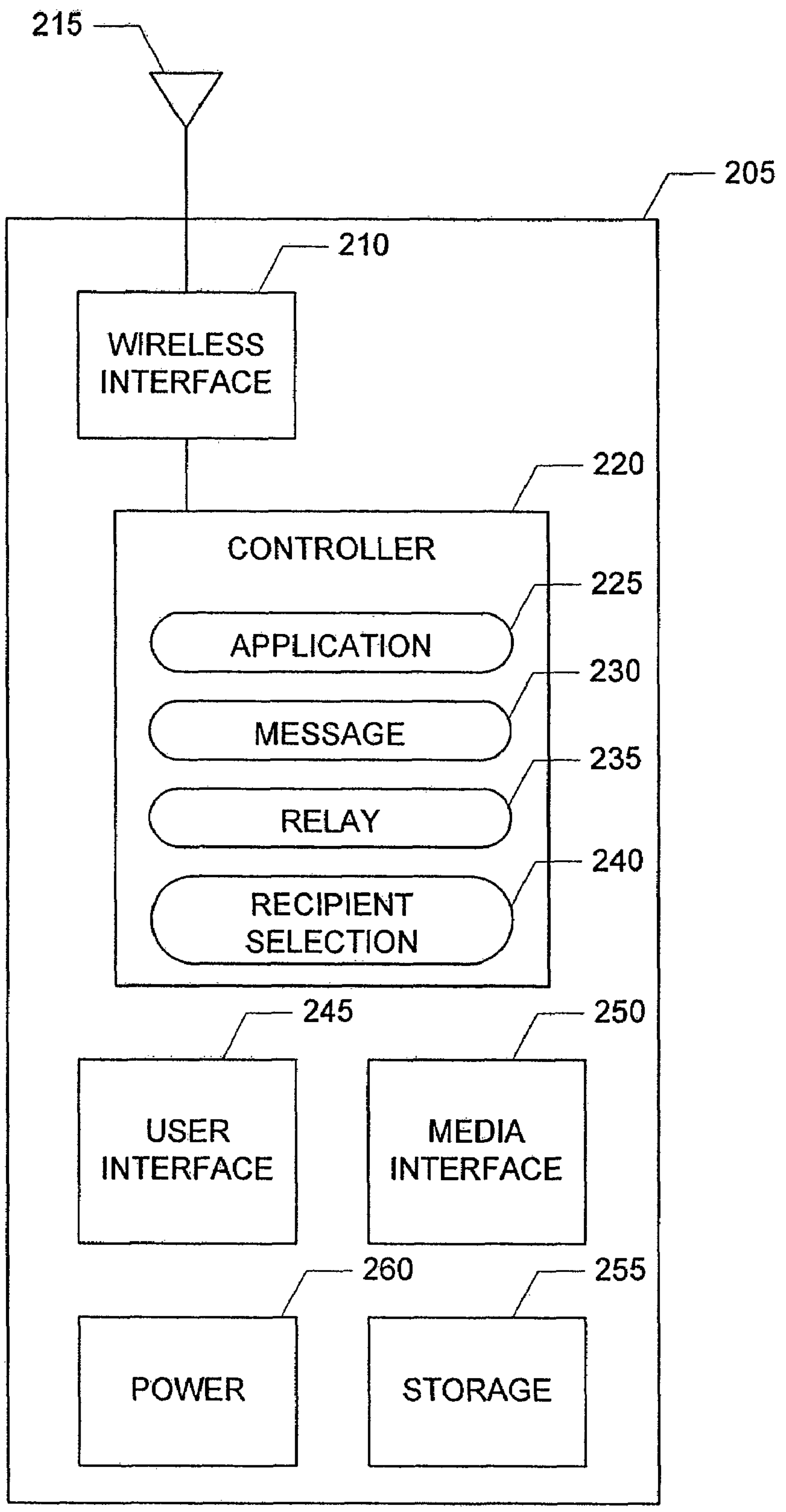

215
205
210
WIRELESS INTERFACE
220
CONTROLLER
225
APPLICATION
230
MESSAGE
235
RELAY
240
RECIPIENT SELECTION
245
250
USER INTERFACE
MEDIA INTERFACE
260
255
POWER
STORAGE

FIG. 3

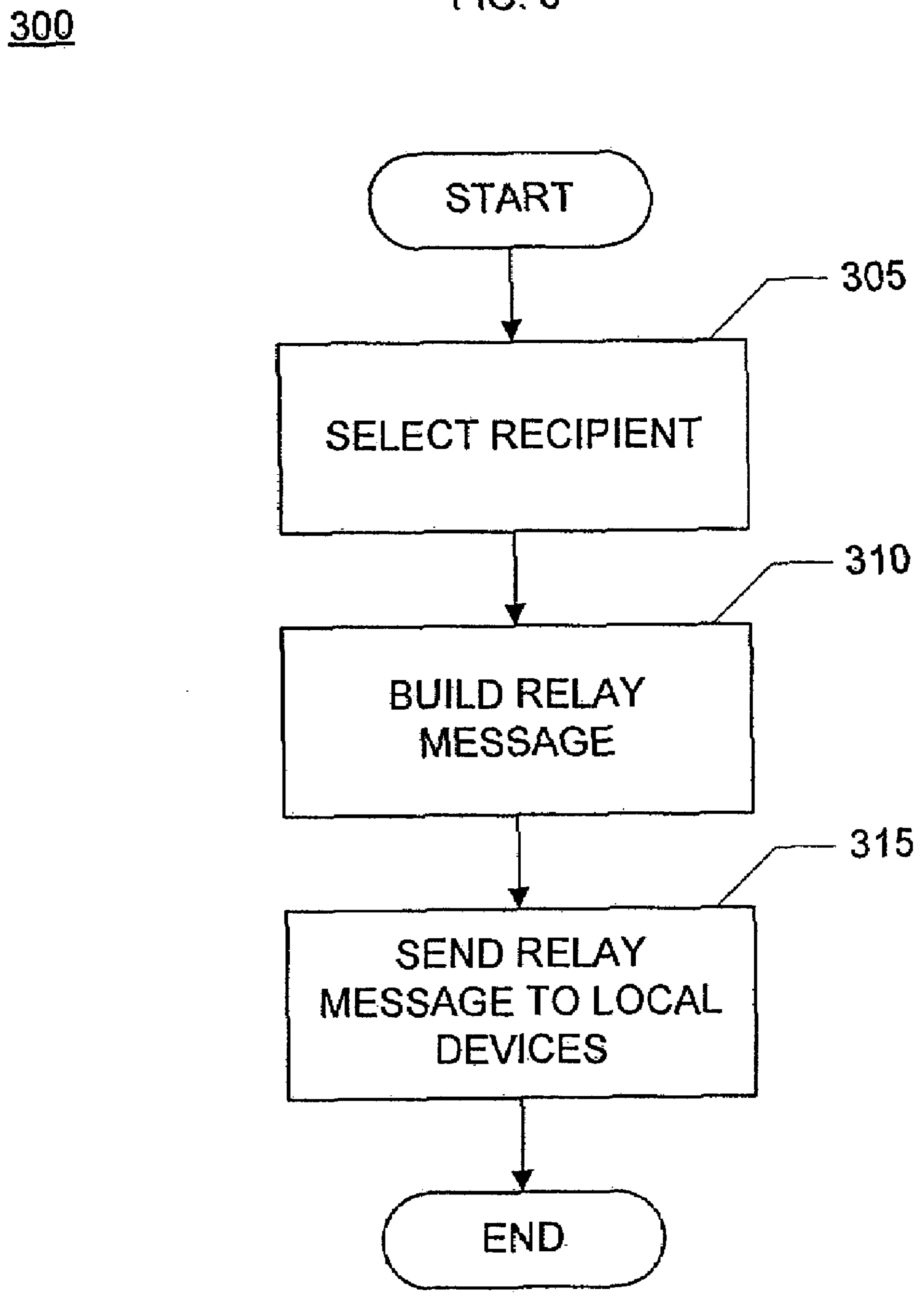

300
START
305
SELECT RECIPIENT
310
BUILD RELAY MESSAGE
315
SEND RELAY MESSAGE TO LOCAL DEVICES
END

FIG. 4

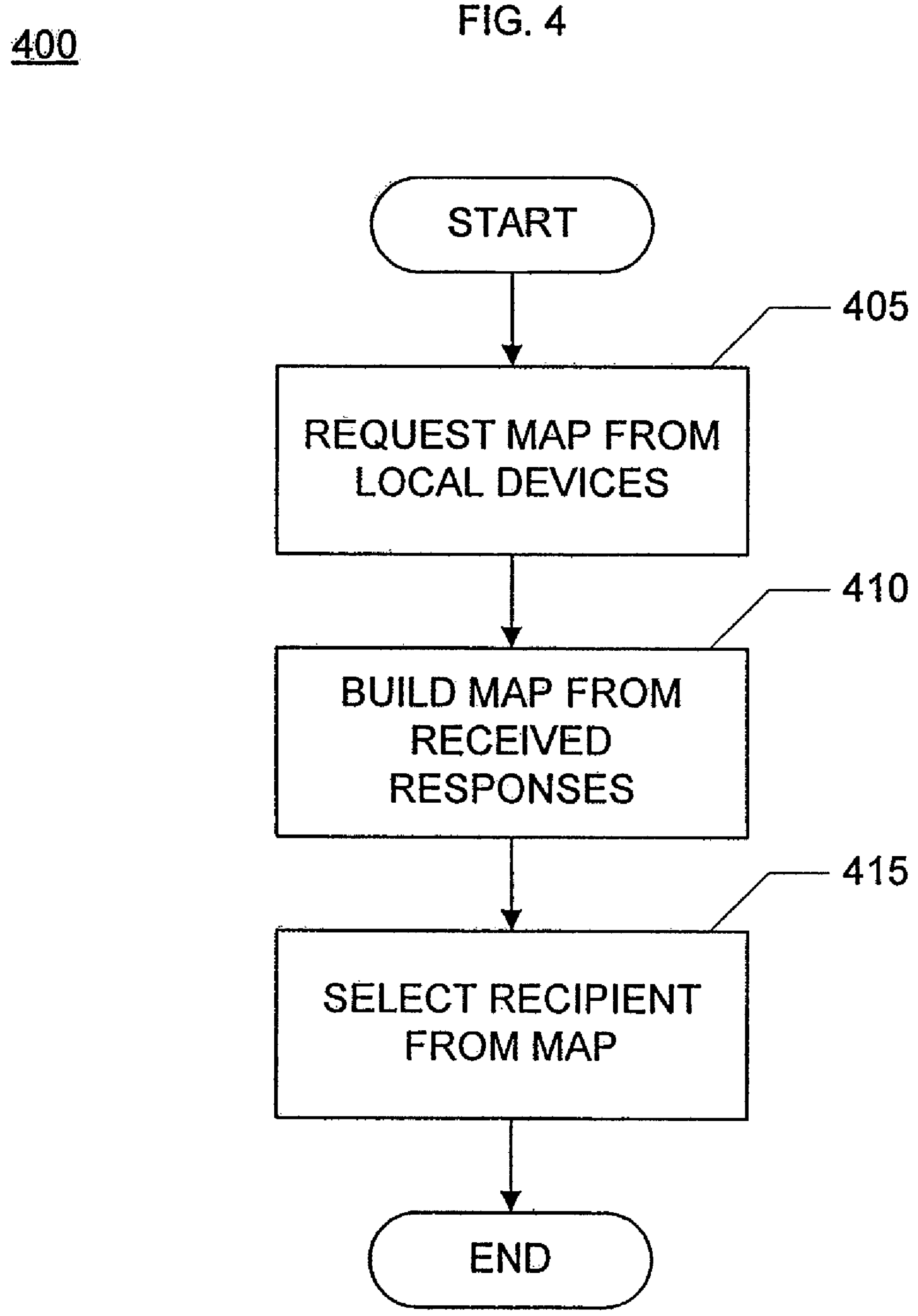

400
START
405
REQUEST MAP FROM LOCAL DEVICES
410
BUILD MAP FROM RECEIVED RESPONSES
415
SELECT RECIPIENT FROM MAP
END

FIG. 5

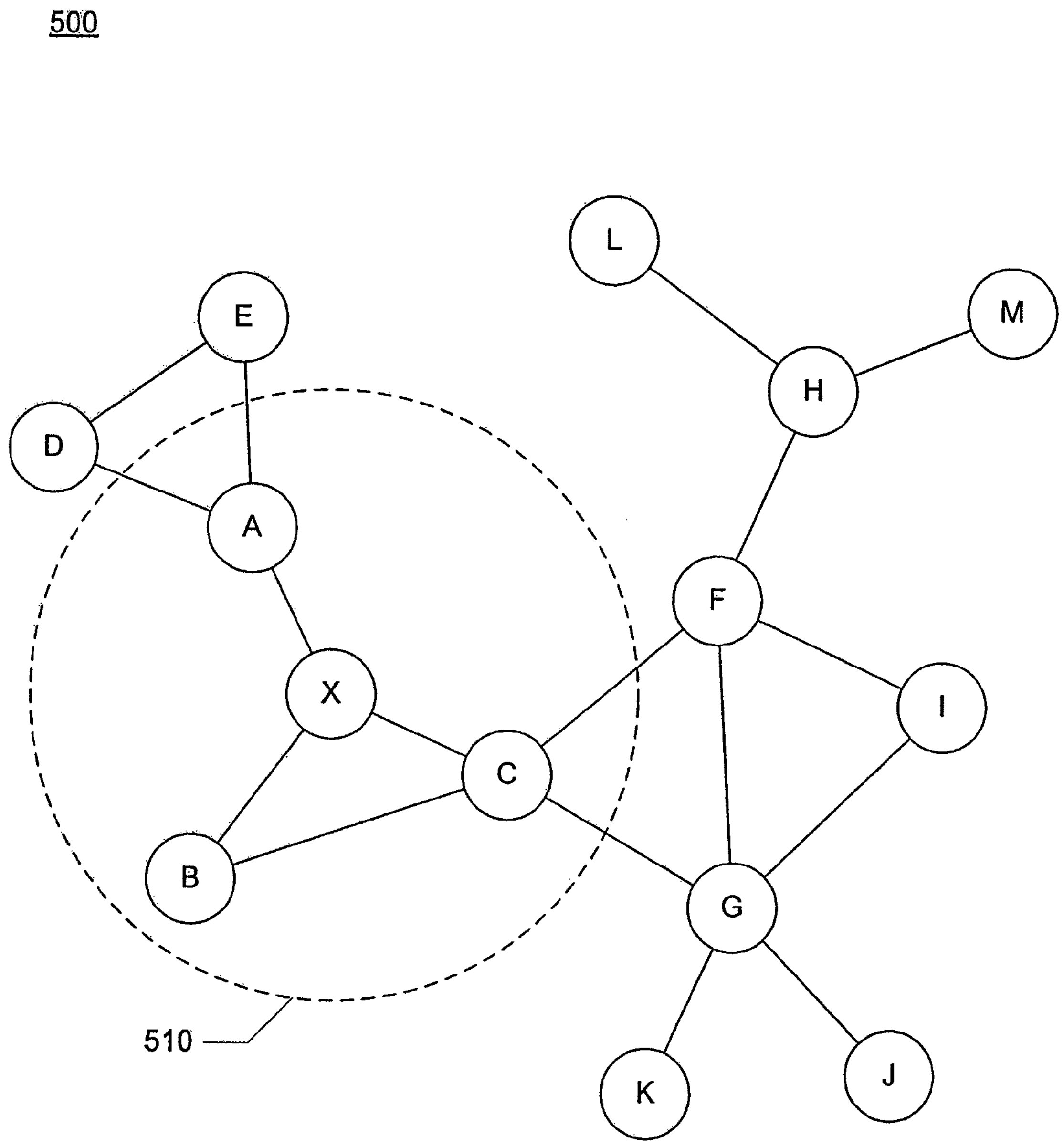
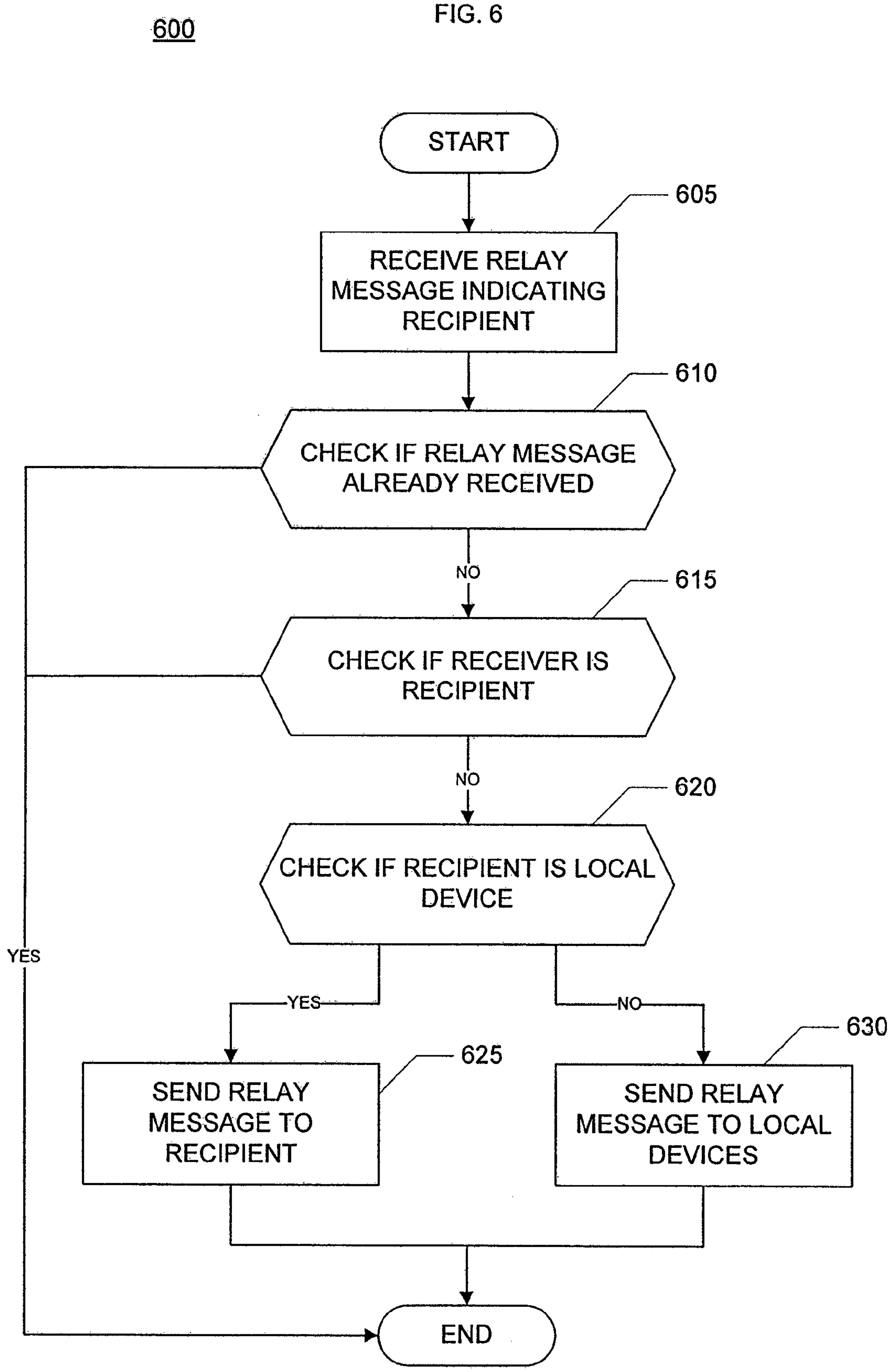


FIG. 6


600
START
605
RECEIVE RELAY MESSAGE INDICATING RECIPIENT
610
CHECK IF RELAY MESSAGE ALREADY RECEIVED
NO
615
CHECK IF RECEIVER IS RECIPIENT
NO
620
CHECK IF RECIPIENT IS LOCAL DEVICE
YES
YES
NO
625
SEND RELAY MESSAGE TO RECIPIENT
630
SEND RELAY MESSAGE TO LOCAL DEVICES
END

PEER-TO-PEER DATA RELAY

This is a division of application Ser. No. 10/970,120, filed Oct. 20, 2004, which is a continuation-in-part of application Ser. No. 10/700,798, filed Nov. 3, 2003, that claims the benefit of provisional application 60/513,098, filed Oct. 20, 2003, the entirety of which is incorporated herein by reference.

BACKGROUND

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server), though the data may pass through other network devices, such as a router or access point. The connections can be wired (e.g., using a network cable to connect to a peer or a network) or wireless (e.g., using an interface supporting an IEEE 802.11 or "Wi-Fi" protocol). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). For example, in a peer-to-peer network with N peers, each peer has N-1 connections to other peers.

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for relaying data in a peer-to-peer network. In one implementation, a wireless device includes: an antenna; a wireless communication interface connected to said antenna and supporting wireless communication across a wireless connection provided by said antenna; storage supporting storing data; and a controller connected to said wireless interface and to said storage, supporting an application service, a message service, and a relay service for relay messages; wherein said application service provides execution and management of one or more application programs accessible by said controller using application data stored in said storage, said message service provides building messages and processing received messages, and said relay service provides building a new relay message indicating a selected recipient to which the wireless device does not have a direct wireless connection, sending a built new relay message, and sending a received relay message that indicates a recipient other than the wireless device.

In another implementation, a method of sending a relay message includes: selecting a recipient device for a relay message at a sending device; building said relay message, wherein said relay message includes a recipient identifier indicating said selected recipient device; sending said relay message to at least one local device through a wireless interface; wherein said sending device has a direct connection to each of said at least one local device, and said sending device does not have a direct connection to said recipient device.

In another implementation, a method of sending a relay message includes: receiving a relay message from a sending device through a wireless interface at a relaying device, wherein said relay message includes a recipient identifier indicating a recipient device and includes application data for an application program; checking said relay message to determine whether said relaying device is said recipient device; checking a local device list indicating at least one local device to determine whether said recipient device is a local device; if said recipient device is a local device, sending said relay message from said relaying device to said recipient device through said wireless interface; and if said recipient device is not a local device, sending said relay message from said relaying device to at least one local device included in said local device list through said wireless interface; wherein said sending device has a direct connection to said relaying device, said relaying device has a direct connection to each local device indicated in said local device list, and said sending device does not have a direct connection to said recipient device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A shows three wireless devices in communication in a peer-to-peer network. FIG. 1B shows another network environment where two wireless devices, a network device, and a server system communicate using direct and relay communication.

FIG. 2 shows a block diagram of one implementation of a wireless device supporting peer-to-peer relay communication.

FIG. 3 shows a flowchart of one implementation of sending a relay message.

FIG. 4 shows a flowchart of one implementation of selecting a recipient using a recipient map.

FIG. 5 shows an example of one network configuration as a graph of nodes and links.

FIG. 6 shows a flowchart of one implementation of sending a relay message from an intermediary or relay device.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for relaying data in a peer-to-peer network. In one implementation, a first wireless device and a second wireless device belong to an extended peer-to-peer network and are exchanging data to support application programs executing on the respective devices. However, the first wireless device and the second wireless devices do not have a direct connection in the peer-to-peer network. The first and second wireless devices have direct connections to one or more other wireless devices (local devices) and can trace a path through intermediary devices to reach one another. To communicate, the first wireless device builds a relay message indicating the second wireless device as the intended recipient of the relay message. The first wireless device sends the relay message to each of its local devices. The local devices receive the relay message and pass the relay message to their respective local devices, following a set of relay rules. The relay message propagates through the peer-to-peer network and reaches the second wireless device. Similarly, the second wireless device sends data to the first wireless device using relay messages.

An illustrative example of one implementation is presented below. This example is not exhaustive and additional examples and variations are also described later.

In one example, a user of a wireless game device brings the wireless game device into an area (e.g., a shop) where an extended wireless peer-to-peer network including other wireless game devices is active. In the extended peer-to-peer network, some of the devices cannot establish direct connections to one another. The wireless game devices support wireless connections of limited range, such as using interfaces supporting the Wi-Fi protocol. For two devices that are out of range, the devices cannot establish a direct connection. However, the two devices can use another intermediary device (or series of devices) as a conduit to send relay messages back and forth.

The user activates his or her wireless game device and requests a list of available devices to play a multiplayer game. The user's wireless game device builds a list of available recipients, indicating each recipient by user name or identifier, or by device identifier. The user's wireless game device builds the recipient list by contacting the other wireless game devices that are within range. These devices within range are local devices for the user's wireless game device. The local devices propagate the list request through the peer-to-peer network and accumulate information indicating the available devices, returning the information to the originator of the list request—the user's wireless game device (examples of building the list are described below).

The user selects a recipient from the recipient list and requests that the wireless game device contact the selected recipient with an invitation to play a game. The user's wireless game device checks whether the selected recipient is a local device or not. If the selected recipient is a local device, a direct connection is available between the user's wireless game device and the recipient and so the user's wireless game device will use direct communication with the selected recipient. The user's wireless game device sends an invitation message to the recipient (not through other local devices) and direct communication proceeds across the direct connection. If the selected recipient is not a local device, a direct connection is not available and so the user's wireless game device will use relay messages for relay communication with the selected recipient.

In relay communication, a wireless game device builds a relay message indicating a selected recipient and including data to be used by the selected recipient. A relay message also includes data to be used by the recipient and a relay message identifier to identify the relay message. To send an invitation to the selected recipient, the wireless game device includes appropriate information in the relay message to indicate the game, the sending device and the user, etc. For relay messages used for the game, the wireless game device includes appropriate application information in the relay messages. The user's wireless game device builds a relay message for an invitation and sends the relay message to each of the wireless game device's local devices through direct wireless connections. None of the local devices are the intended recipient for the relay message, and so these local devices are intermediary or relay devices in this relay communication. When an intermediary device receives the relay message, the intermediary device checks whether the indicated recipient is a local device of the intermediary device, using a list of its local devices. If the recipient is a local device, the intermediary device sends the relay message to the recipient. If the recipient is not a local device, the intermediary device sends the relay messages to all of its local devices according to a set of relay rules. In this example, one relay rule indicates that an intermediary device does not send a relay message back to the local device from which the relay message was received. Another rule indicates that an intermediary device does not send the same relay message again (indicated by the same relay message identifier). After sending the relay message, the intermediary device does not retain the relay message, but does record the relay message identifier to avoid re-sending the same relay message.

The relay message propagates through the peer-to-peer network and the wireless game device indicated as the recipient eventually receives the relay message. The recipient recognizes that it is the recipient and uses the included data as appropriate. When the recipient device receives the invitation from the user's wireless game device, the recipient device presents the invitation to the recipient's user and accepts appropriate instructions. The recipient device builds a relay message indicating the user's wireless game device as the recipient of this relay message and including information indicating the response to the invitation. The recipient device sends the relay message back through the peer-to-peer network to the user's wireless game device. The two devices can then continue to communicate in the same way using relay communication.

In this example, two wireless devices that do not have a direct wireless connection use other wireless devices as intermediary devices to create an indirect or relay connection. By using relay communication, the wireless devices advantageously can overcome limitations in range and expand the area and devices available for communication. Furthermore, the two wireless devices communicate through a series of peer-to-peer connections without requiring an intermediary server system. The two wireless devices can exchange data to facilitate the joint operation of application programs (e.g., a multi-player game application) without any of the intermediary devices being participants in the joint operations. For example, two wireless devices can jointly play one game using a third wireless device as a relay, while that third device is playing some other game with yet another device. Different implementations and applications can realize additional or different advantages as well.

FIG. **1**A shows three wireless devices **105**, **110**, and **115** having respective wireless antennas **107**, **112**, and **117** in communication in a peer-to-peer network. The wireless devices **105**, **110**, **115** communicate as peers rather than through a server. In one implementation, the wireless devices **105**, **110**, **115** are wireless-enabled game or general-purpose computing devices (e.g., game devices, PDA's, or laptop computers supporting Wi-Fi interfaces, or cellular phones, etc.). Using its antenna **107**, the wireless device **105** establishes a direct wireless connection with the wireless device **110**. Similarly, the wireless devices **110** and **115** establish a direct wireless connection. In the configuration shown in FIG. **1**A, the wireless device **105** and the wireless device **115** do not have a direct connection, such as because they are out of range for their wireless interfaces.

The wireless devices **105**, **110**, **115** exchange data (e.g., messages) across the wireless connections. The wireless device **105** communicates directly with the wireless device **110**. The wireless device **110** communicates directly with both of wireless devices **105** and **115**. The wireless device **115** communicates directly with the wireless device **110**. As described below, the wireless device **105** can communicate indirectly with the wireless device **115** using relay communication through the wireless device **110** as an intermediary or relay device. In this way, the wireless device **110** can assist as a relay while functioning as a peer in the extended network.

FIG. **1**B shows another network environment where two wireless devices **120**, **125**, a network device **130**, and a server system **135** communicate using direct and relay communication. The wireless devices **120** and **125** establish a direct wireless connection using respective antennas **122** and **127**. The wireless device **125** also communicates across a wireless connection with a network **140**. The network device **130** and server **135** communicate with the network **140** through wired connections using respective network interfaces **132** and **137**. In one implementation, the network device **130** is a network-enabled computing device, such as a laptop computer or a game console having a network interface. In one implementation, the server **135** is a computing system providing data and/or application serving to connecting clients (such as the wireless devices **120**, **125** and the network device **130**). In one implementation, the network **140** is a local area network (LAN). In another implementation, the network **140** is a wider network, such as the Internet or a private intranet.

The wireless device **125** can establish direct connections to the network device **130** and the server system **135** though the network **140** (though data may pass through other network equipment, such as a base station and routers). In this configuration, a 'direct' connection is distinguished from an 'indirect' connection in that a direct connection is a peer-to-peer connection and so a first device sends a message addressed to a second device (e.g., using TCP/IP) across a direct connection to the second device. An indirect connection includes a series of two or more direct connections. For example, the wireless device **125** has a direct connection to the wireless device **125** and a direct connection to the network device **130**. Accordingly, when the wireless device **125** sends messages to the network device **130**, those messages are addressed to the network device **130** as a peer. The wireless device **120** has an indirect connection to the network device **130**, through a first direct connection between the wireless device **120** and the wireless device **125** and a second direct connection between the wireless device **125** and the network device **130**. When the wireless device **120** sends messages to the network device **130**, those messages are relay messages addressed to the wireless device **125** and indicating the network device **130** as the intended recipient. In turn, the wireless device **125** relays to the network device **130** those relay messages from the wireless device **120** by sending to the network device **130** new relay messages that include the same payload data (e.g., non-addressing data) as the received relay messages but are addressed to the network device **130** using the direct connection. Alternatively, the wireless device **125** sends the same relay messages but changes the addressing information.

In one implementation, the server **135** provides centralized storage of information and so the devices provide and request information to and from the server **135** (e.g., periodically or upon demand). For example, in one implementation, the server **135** stores a list of the devices participating in the peer-to-peer network. In another example, the server **135** stores a list of local devices for each participant in the peer-to-peer network. Alternatively, the server **135** provides different information that complements the communication among the devices. For example, the server **135** stores user profiles for users of devices indexed by device identifiers.

The wireless devices **120**, **125**, the network device **130**, and the server **135** can have flexible roles. Each can act as a peer, a client, or a server as needed. For example, the server **135** can meet a centralized storage request for the wireless device **120** and then interact as a peer with the wireless device **125**.

While FIGS. **1**A and **1**B illustrate two network configurations, in other implementations, other network configurations can be used (see, e.g., FIG. **5**). For example, in one implementation, a collection of wired network devices exchange information through a LAN or wider network, without using wireless connections. In one implementation using wired devices, a participant limit is used to control the number of devices participating in a peer-to-peer network and a local device limit is used to control the number of devices that a device considers as local devices (available for direct connections). In another example, various types of wireless devices are used, such as cellular phones and PDA's.

FIG. **2** shows a block diagram of one implementation of a wireless device **205** supporting peer-to-peer relay communication. The wireless device **205** includes a wireless interface **210** connected to an antenna **215**. The wireless interface **210** supports establishing a wireless connection to another wireless device using the antenna **215** and exchanging data across an open connection. In one implementation, the wireless interface **210** is a typical radio interface supporting an air interface, such as a local wireless protocol like Wi-Fi or Bluetooth. While the components of the wireless interface **210** are not shown in FIG. **2**, those components and their operation will be understood by one of ordinary skill in the art (e.g., the appropriate filters, amplifiers, etc.). In another implementation, a different air interface can be supported, such as CDMA, or a non-radio interface, such as infrared. In another implementation, multiple wireless interfaces are supported and/or one or more wired interfaces (e.g., IEEE-1394 or Ethernet).

A controller **220** is connected to the wireless interface **210**. The controller **220** controls the operation of the wireless device **205** and its components. In one implementation, the controller **220** is a microprocessor and related sub-systems. The controller **220** provides an application service **225**, a message service **230**, a relay service **235**, and a recipient selection service **240**. The services of the controller **220** are implemented as software programs stored in storage and executed by the controller **220**. Alternatively, one or more of the services can be implemented partially or completely as hardware or in a separate sub-system of the wireless device **205**. In another implementation, the controller does not provide any of these services and instead the operation of the wireless device is controlled manually by a user through the user interface of the wireless device (e.g., the controller controls the operation of the components according to the user commands as a control processor rather than providing services for automatic operation for sending messages etc.).

The application service **225** supports the execution of application programs accessible to the wireless device **205** and management of corresponding application data. In one implementation, the application service **225** supports executing game application programs.

The message service **230** supports building messages, sending messages through the wireless interface **210**, and processing messages received through the wireless interface **210**. The message service is primarily for messages used in direct communication.

The relay service **235** supports building relay messages, sending relay messages, and processing received relay messages, as described below (e.g., referring to FIGS. **3** and **6**). The relay service **235** is primarily for messages used in relay communication and operates in conjunction with the message service (e.g., sharing sub-services such as address insertion or lower protocol layer functions). In one implementation, the relay service **235** builds and maintains a local device list indicating the devices with which the wireless device **205** can establish a direct connection.

The recipient selection service **240** supports building a list or map of available recipients for communication and selecting a recipient for communication, as described below (e.g., referring to FIGS. **4** and **5**). In one implementation, the recipient list is a list of devices, with corresponding addressing information for local devices (as indicated by the local device list). The recipient list can also indicate (or reference) additional information about the recipients (e.g., user name). In another implementation, the recipient selection service **240** builds a map of available recipients, indicating both the available recipients and for each recipient that is not a local device indicating one or more intermediary devices (or connections) used to reach that recipient. The recipient selection service

240 builds the list or map by querying the local devices. Building the recipient list or map is described further below. In another implementation, the recipient selection service **240** builds a list or map using information received from a server, or obtains a list or map from a server.

The wireless device **205** also includes a user interface **245**, a media interface **250**, storage **255**, and a power source **260**. The user interface **245** provides input controls to receive user commands (e.g., a keypad, buttons, a directional pad or joystick, etc.) and output components to provide data to the user (e.g., a display and audio speakers). The media interface **250** provides components to connect to or accept media to exchange data with the media, according to the type of media. In one implementation, the media interface **250** provides an interface to removably receive an optical disc to read data from the disc (e.g., a CD-ROM drive) and another interface to connect to a removable memory component to read data from and write data to the memory component (e.g., a USB port to accept a USB memory card, a removable hard disk drive interface, or a Memory Stick™ interface). The storage **255** provides storage for data used in the operation of the wireless device **205** and in application execution. For example, in one implementation, relay messages and recipient lists are stored in the storage **255**. In one implementation, the storage **255** is a combination of RAM and flash memory. In another implementation, the storage **255** also provides storage for data over a longer period, such as user data and application data and files, and includes appropriate long-term storage, such as flash memory and/or a hard disk drive. The power source **260** provides power to the components of the wireless device **205**. In one implementation, the power source **260** provides an interface for drawing power from removable batteries and a rechargeable internal power source with a corresponding external power connection. In FIG. **2**, the interconnections among the user interface **245**, media interface **250**, storage **255**, and power source **260**, and with the other components of the wireless device **205** are not shown for clarity, though these connections will be appreciated by one of ordinary skill.

In another implementation, a network device that is not wireless-enabled but does support an interface for communication with other devices provides similar components to the wireless device shown in FIG. **2**. The network device includes a controller providing application, message, relay, and recipient selection services, and so also support relay communication as described below. As described above referring to FIG. **1**B, a combination of wireless devices and network devices can be used for relay communication.

FIG. **3** shows a flowchart **300** of one implementation of sending a relay message. Initially, a wireless device supporting relay communication is participating in a peer-to-peer network. The wireless device has built a list of local devices (e.g., using a beacon) and opened direct connections to those local devices. The wireless device receives a request from a user to select a recipient.

The wireless device selects a recipient for communication, block **305**. The wireless device builds a recipient map indicating available recipients and corresponding paths of connections, such as by using the techniques described below referring to FIG. **4**. In another implementation, the wireless device builds a list without indicating connections, rather than a map. In one implementation, the wireless device does not build a new recipient map (or list) if the wireless device already has a recipient map and that map is current (e.g., as determined using an expiration threshold and a creation or last update timestamp of the map, or by querying local devices). In another implementation, the wireless device selects a recipient from a list or map provided by a server, or selects a recipient through a server (and optionally selecting or receiving from the server a path to the recipient).

The wireless device presents the recipient map to the user and receives a selection choice indicating a recipient. The wireless device determines whether the selected recipient is a local device or not by checking the local device list. In another implementation, the recipient list also indicates whether each recipient is a local device or not (for the wireless device storing the particular recipient list). If the selected recipient is a local device, the wireless device uses direct communication to exchange data with the selected recipient. If the selected device is not a local device, the wireless device will use relay communication.

After selecting a recipient that is not a local device, the wireless device builds a relay message, block **310**. The relay message includes information indicating the selected recipient, such as a recipient identifier from the recipient list. In one implementation where the recipient list indicates a full or partial path to the recipient, the wireless device includes the path information in the relay message. In another implementation, after selecting a recipient from a list, the wireless device discovers the path to the selected device and includes the path information in the relay message. In another implementation, the relay message includes information indicating multiple recipients (e.g., a list). The relay message includes a relay message code to identify the relay message as a relay message. The wireless device assigns a relay message identifier to the relay message to identify the relay message as the message is sent through the peer-to-peer network (to identify the relay message relative to other relay messages). The wireless device includes the relay message identifier in the relay message. In one implementation, the wireless device also includes a hop count limit to limit how far through the network the relay message should propagate (hop counts are described further below). The wireless device generates appropriate additional data to be sent to the recipient and includes that data as the payload data of the relay message. For example, when the wireless device is going to play a game with the recipient device, the wireless device includes appropriate game data in the relay message.

The wireless device sends the built relay message, block **315**. In one implementation, the wireless device broadcasts the relay message through the wireless interface of the wireless device, sending the relay message to all of the devices in the local device list of the wireless device. In another implementation, the wireless device sends the relay message to one or selected local devices according to path information in the relay message.

After the wireless device has opened relay communication with a recipient, the wireless device continues to build and send relay messages to the recipient as described above referring to blocks **310** and **315**.

Wired devices or wireless devices including wired interfaces use similar techniques to send relay messages, selecting the appropriate interface to send relay messages for corresponding local devices.

In one implementation, the wireless device has a maximum peer count to limit the number of local devices to present as available for selection as recipients or to use as intermediary devices. The wireless device builds a first local device list indicating all the local devices available (e.g., determined by responses to a beacon). The wireless device builds a second local device list by selecting a number of devices equal to or less than the maximum peer count from the available local devices. The wireless device can use various criteria to select devices for the second list, such as response time or signal strength. In one implementation, the maximum peer count value is dynamic. The wireless device adjusts the maximum peer count based on the wireless device's current performance load (e.g., increasing the limit when the device is idle) or based on network traffic (e.g., reducing the limit when traffic is very high). In another implementation, the wireless device sets the maximum peer count based on information from the user or from a server.

FIG. **4** shows a flowchart **400** of one implementation of selecting a recipient using a recipient map, such as to select a recipient in block **305** of FIG. **3**. Initially, a wireless device supporting relay communication is participating in a peer-to-peer network. The wireless device has built a list of local devices (e.g., using a beacon) and opened direct connections to those local devices. The wireless device has received a request from a user to select a recipient.

The wireless device requests a recipient map from each of the local devices, block **405**. A recipient map indicates one or more devices and a full or partial path for each of the devices that is not local to the owner of the map. The owner of the map is the device that built the map and upon which the paths are based. Accordingly, a path indicates one or more corresponding connections or intermediary devices between the owner of the map and the particular recipient for that path. A full path shows all the intervening connections or devices and a partial path shows one or more of the intervening connections or devices, such as the final one. In one implementation, a recipient map is a graph, with nodes indicating devices and links indicating connections. In one implementation using a graph, the map does not indicate an owner. In another implementation, a recipient map is a table, with one entry for each recipient and a field indicating a path from the owner to the corresponding recipient. The local devices return their recipient maps to the wireless device. In one implementation, the local devices update their maps before returning them. In one implementation, the map request is a request for a map or alternatively to build a new map if the current map is too old. In another implementation, the map request is sent as a broadcast, rather than to specific local devices. Examples of building maps are described further below.

The wireless device builds a recipient map based on the received responses to the map requests, block **410**. The wireless device combines the received maps and its own local device list to build a map with the wireless device as the owner (so that paths are defined relative to the wireless device). In another implementation, the wireless device does not send a map request or build a new recipient map if the wireless device already has a recipient map and that map is current (e.g., as determined using an expiration threshold and a creation or last update timestamp of the map).

The wireless device selects a recipient from the map, block **415**. The wireless device presents the map to the user through the user interface of the wireless device. The user selects a recipient and the wireless device selects a recipient corresponding to the user selection.

Various techniques can be used to build recipient maps or lists (the techniques may apply to either or both of list and maps, as appropriate). Several examples are described below, including: propagation, hop count, network identifier, and periodic updating. Combinations of these or other building techniques can also be used.

In one implementation, a device updates its recipient map using one or more update rules. One rule is to update the recipient map upon joining the environment. Another rule is to update the recipient map at regular intervals. The device sends out a map request when the recipient map has expired, based on a comparison of the elapsed time since the last update of the recipient map. Another rule is to update the recipient map when a local device becomes unavailable. When the device detects that a local device is no longer available (e.g., based on regular polling or a failure to respond to a request), the device sends out a map request. Another rule is to update the recipient map when a new local device becomes available. Another rule is to update the recipient map when the device receives a refresh message from a local device indicating that local device has updated its recipient map. When a device has changed its recipient map, the device sends a refresh message to each of its local devices. The devices receiving the refresh message then each send out map requests to update their recipient maps. One or more of these or various other rules can be used to build, update, and maintain recipient maps.

FIG. **5** shows an example of one network configuration **500** as a graph of nodes and links. Example of map building are described below referring to the configuration of FIG. **5**. Each node represents a device and each link between nodes represents a direct connection between two devices. Each of the nodes are labeled with a letter for the corresponding device. For example, a device X corresponds to the node labeled X. In FIG. **5**, the device X has direct connections with three devices: A, B, and C. The device X is a wireless device and the limit of the range of the device X is indicated by the dashed circle **510** centered on the node X. The device X cannot establish direct wireless connections with devices outside its range. For example, the device D is outside the range of the device X (as indicated by the node D being outside the dashed circle **510**). Similarly, the other devices have also established direct connections with the devices that are in range. In another configuration, some or all of the devices are wired devices and limit establishing direct connections on some other basis (e.g., an artificial limit such as a local device limit set by a user).

Propagation

In one implementation of building a map, the devices in the peer-to-peer network use propagation to build and update maps. Initially, a device builds a map request and sends the map request to each of its local devices. A map request requests a map of available devices and has an identifier to identify the request. Each of the local devices receiving the map request in turn sends a map request to that device's local devices using the same map request identifier, but does not send a map request back to the sender of the map request and does not send a map request when the device has already sent a map request having the same identifier. The devices receiving the map request follow a similar process to send out map requests. To send a map back to the sender of the request, a device waits for a response from each of the local devices to which the device sent the map request. If a device receives a map request with the same identifier as a previously received map request, the device sends back a failure message. If a device has no available local devices to which to send the map request (e.g., the only local device is the sender of the map request), the device returns the device's local device list to the sender of the map request or builds and returns a map based on the local device list. After receiving responses from each of the local devices to which the device sent the map request (discarding failure messages), the device combines the information of the received maps (if any) with the device's local device list to build (or update) its map. The device then returns the new map to the sender of the map request. This process continues until the original sender of the map request has received maps from its local devices and built its own map.

Referring to FIG. **5**, in an example of building a map using the propagation technique described above, the device X sends a map request to each of its local devices: to the devices A, B, and C. The device A sends a map request to its local devices, except for the device X as the sender of the map request: to the devices D and E. The map request sent by device A includes the same map request identifier as that in the map request sent by the device X, and so will all the map requests propagated from the original map request sent by the device X. The device D receives the request from the device A and sends a map request to the device E (not to the device A because the device D received the map request from the device A). Similarly, the device E sends a map request to the device D. When the device D receives the map request from the device E, the device D recognizes that this map request is the same as the map request the device D received from the device A and then sent to the device E (by matching map request identifiers). Accordingly, the device D sends a failure message (or some message indicating the map request has already been sent) back to the device E. Similarly, the device E sends a failure message back to the device D. When the device D receives the failure message from the device E, the device D recognizes that there are no more pending responses (because the device D sent the map request only to device E). The device D has not received any maps as responses and so sends back its local device list (indicating devices A and E) to the device A as the response to the map request from the device A. Similarly, the device E returns its local device list to the device A (indicating the devices D and A). The device A recognizes it has received all its pending responses and builds a map from the received responses and its local device list. The map the device A builds indicates: the device D is available as a recipient through a connection from the device A to the device D, the device E is available as a recipient through a connection from the device A to the device E, and the device A is available as a recipient (e.g., as D-A, E-A, and A). When two maps provide alternate paths to the same device, the shorter path is used (measured by more intermediary devices being considered longer). When two paths are the same length, an arbitrary selection is made, or both paths are included. Accordingly, the device A does not indicate in its map to return to the device X that a path is available from the device D to the device E because there is a direct path from the device A to the device E.

The device B receives the map request from the device X and sends a map request to the device C. However, the device C has already received the map request from the device X and so sends a failure message back to the device B. The device B has no more pending responses and has not received any maps, so the device B returns its local device list to the device X: the devices X and C. Alternatively, rather than sending back the local device list, the device B returns a map indicating: the device C is available as a recipient through a connection from the device B to the device C, and the device B is available as a recipient (e.g., C-B and B).

The device C receives the map request from the device X and sends a map request to its local devices except the device X: the devices B, F, and G. The device B returns a failure message. The device F in turn sends a map request to its local devices: the devices H, I, and G. The device G sends a map request to its local devices: the devices F, I, J, and K. The devices continue to propagate the map requests, and build and return maps as described above. When the device C has received its pending responses to the send map requests, the device C builds a map indicating (using the example notation described above): L-H-F-C, M-H-G-C, H-F-C, I-F-C, J-G-C, K-G-C, F-C, G-C, and C.

When the device X has received the pending responses from the devices A, B, and C, the device X combines the

responses and its local device list to build a map indicating these recipients and paths: A, B, C, D-A, E-A, F-C, G-C, H-F-C, I-F-C, J-G-C, K-G-C, L-H-F-C, and M-H-G-C.

Using this map, the device can present the available recipients in various ways: as a graphical map (e.g., showing connections), as a list or recipients, as a list with a distance indicator (e.g., number of connections to reach or color code). If the user selects the device H as a recipient, for example, the device X can include in the relay message to the device H the path to the device H: X to C to F to H. When the device X sends the relay message, the device X uses the path information and sends the relay message to the device C, and not to the other local devices A and B. Similarly, the device C passes the relay message to the device F (not B or G), and so on.

In another implementation, the wireless device uses a recipient list instead of a map. A recipient list indicates the available recipients, but does not indicate path information (explicitly). One type of recipient list does indicate whether a recipient is a local device or not, relative to the owner of the list. Using the propagation technique described above, the devices return lists rather than maps and the device X builds a list indicating the devices: A, B, C, D, E, F, G, H, I, J, K, L, and M.

In another implementation, the device maintains redundant path information. The device stores a recipient map indicating available paths and applies a filter to remove redundant paths when presenting the map for recipient selection. If another device in the map becomes unavailable, the device can use the redundant path information to identify other paths to devices that were shown as being connected through the unavailable device. For example, in FIG. **5**, there are two paths from the device I to the device X: I-F-C-X and I-G-C-X. If the filter of the device X has selected the first path through the device F for display in recipient selection, the second path through the device G is redundant and is not displayed. If the device F becomes unavailable, the device X can display the second path through the device G as available because the device X has maintained the redundant path information.

Hop Count

In another implementation, the devices use a hop count as a limit of how far out (across how many connections) to propagate the map request or list request. Various definitions of "hop" can be used. In one definition, a hop is a connection or link between two devices. Referring to the configuration shown in FIG. **5**, the connection between the device X and the device C is 1 hop. In the path between the device X and the device H, there are 3 hops. In another definition, a hop is a connection between two intermediary devices along a path. Referring to FIG. **5**, in the path between the device X and the device H, there is 1 hop. The connection between the device X and the device C is not a hop because the device X is not an intermediary device. The connection between the device C and the device F is a hop because the devices C and F are intermediary devices. The connection between the devices F and H is not a hop because the device H is the recipient and not an intermediary device. Accordingly, the hop count for the path between the device X and the device H is 1. In other implementations, different definitions of hop count can be used, such as every connection after the first being counted as a hop (in which case the hop count between the devices X and H would be 2).

When the hop count reaches a threshold, a device will not send another map or list request. In an example using list requests and where the hop count is limited to 3, a device will not send a list request when the hop count has reached 3. This limit of 3 will build a list where all of the recipients are 3 hops or less away. The hop count is included in the list request. Using a definition of hops where each link is a hop, the hop count is initially 1 and is incremented by 1 when a device sends the list request to another device (e.g., incremented by the sender). Referring again to FIG. **5**, when the device X sends the list request to the device C, the hop count is incremented by 1 to be 2. When the device C sends the list request to the device F, the hop count increases to 3. The hop count has reached the limit of 3. Accordingly, the device F does not send the list request to its local devices (G, H, I), and instead returns its local device list to the device C. Following that pattern, when the device X has received its pending responses from the list requests sent out to the devices A, B, and C, the device X builds a list indicating: A, B, C, D, E, F, G, H, I, J, and K. The list does not include the devices L and M because the device H did not receive a list request.

Network Identifier

In another example of controlling the building of recipient maps or lists, the requests include a network identifier. Each of the devices participating in the network stores the same network identifier. When a device sends a map or list request, the device includes that network identifier in the request. If a device receives a map or list request indicating a network identifier different from that of the device, the device returns a decline message to the sender indicating the device is not a participant in that network. When a device receives that decline message, the device does not include the device sending the decline message in the list or map of available recipients. Network participation can be managed through a server or through self-regulation of participating devices (e.g., by invitation or request and acceptance).

Periodic Updating

In another implementation of building a recipient map or list, a device uses periodic updating to build a map or list by periodically requesting a list from each of its local devices (or on demand). A device does not propagate the request. A device returns its current recipient list to the requesting device. When a device receives the responses, the device updates its recipient list by combining the responses with its local device list. When a device later receives a list request from a local device, the device will return the updated recipient list. In this way the recipient lists for each of the devices gradually develop and include all the available devices. A similar approach can be used with maps. Using the path information in a map, changes in available devices can be applied to the map. Referring again to FIG. **5**, if the device X has a complete recipient list (including devices A-M) and receives an update indicating that device C is no longer available, the device X can use the path information to determine that devices F-M are also not available because the device C provided the link to those devices. In one implementation, removing connected devices can prompt a device to send out a new map request (e.g., using a propagation technique).

Various other techniques can be used to build and update recipient maps and lists. For example, in another approach, wireless devices periodically broadcast their local device lists or recipient lists (or maps) and other devices update their list and maps accordingly. In another example, a server provides lists, maps, and/or paths, though communication between devices remains peer-to-peer (direct or relay).

FIG. **6** shows a flowchart **600** of one implementation of sending a relay message from an intermediary or relay device. Initially, a wireless device supporting relay communication is participating in a peer-to-peer network. The wireless device may or may not be executing an application program. The

wireless device has built a list of local devices (e.g., using a beacon) and opened direct connections to those local devices.

The wireless device receives a relay message from one of the wireless device's local devices, block **605**. The wireless device receives the relay message through the wireless interface of the wireless device. The wireless device recognizes that the relay message is a relay message because the relay message includes a relay message code. For relay messages including application program data, the wireless device is not necessarily executing a local copy or version of the application program corresponding to the application data in the relay message, and may be executing an application program different from the application program corresponding to the application data included in the relay message. When appropriate, however, the wireless device can store or use application data in the relay message (e.g., if the wireless device is also executing that application program or a related application program).

The wireless device checks whether the relay message has already been received by the wireless device, block **610**. The wireless device retrieves the relay message identifier from the relay message and compares the retrieved identifier to a received relay message list of stored relay message identifiers (stored from previously received relay messages). If the identifier matches an identifier in the stored received relay message list, the wireless determines that the relay message has been received previously and discards the relay message. If the identifier does not match any in the list, the wireless device determines that this relay message is a new relay message and adds the relay message identifier to the received relay message list.

If the relay message is a new relay message, the wireless device checks whether the wireless device is the indicated recipient for the relay message, block **615**. The wireless device retrieves the recipient information from the relay message and compares that recipient information with information identifying the wireless device. If the wireless device is the indicated recipient, the wireless device does not relay the relay message and proceeds with accessing and processing the payload data of the relay message (e.g., using the payload data as application data for an executing game application program). If the relay message indicates multiple recipients (e.g., the wireless device and at least one additional device), the wireless device processes the payload data and sends the relay message on, proceeding to block **620**.

If the wireless device is not the indicated recipient for the relay message, the wireless device checks whether the indicated recipient is a local device for the wireless device, block **620**. The wireless device compares the recipient information with the wireless device's local device list.

If the recipient is one of the local devices, the wireless device sends the relay message to the recipient through the wireless interface, block **625**. If the recipient is not one of the local devices, the wireless devices sends the relay message to each of the local devices, except for the local device that sent the relay message to the wireless device, block **630**. In an implementation using paths, if the relay message includes path information indicating the next intermediary device and so to which local device the wireless is to send the relay message, the wireless message sends the relay message to the indicated intermediary device.

The process described referring to FIG. **6** follows one set of relay rules (e.g., do not send the same relay message twice). In other implementations using a different set of rules (or only one rule or no rules), the process for determining how to handle a relay message can be different. For example, in one implementation, before sending a relay message on, the intermediary device checks an updated map or list of recipients to confirm that the indicated recipient is still available. In another example, if an intermediary device has not already sent the same relay message, the relay message is always broadcast (e.g., not to specific devices, but to all devices in range).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Some implementations include one or more computer-programs executed by a programmable processor or computer. For example, referring to FIG. **1**A, in one implementation, the wireless device **105** includes one or more programmable processors (e.g., implementing the controller **220** of FIG. **2**). In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description focuses on implementations using wireless devices, wired or multi-interface devices (e.g., wired and wireless) can be used.

In addition, rather than exchanging relay message to support game application programs, other application programs can also be supported, such as such as scientific, communications, etc. In that case, the payload data of the relay messages would reflect the nature of the application programs being addressed (e.g., most recent network test results for network analysis application programs).

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A wireless device, comprising:

an antenna;

a wireless communication interface connected to said antenna and supporting wireless communication across a wireless connection provided by said antenna, the wireless communication interface enabled to contact local devices, which are devices that are within range of a sending device through the wireless interface, and requesting identification of available devices;

storage supporting storing data; and

a controller connected to said wireless interface and to said storage, supporting an application service, a message service, and a relay service for relay messages, the controller enabled to (1) propagate by the local devices the request for identification of available devices through a peer-to-peer network, (2) receive the identification of available devices to the sending device, and (3) build a list of available devices, the list including identification of said local devices in a local device list;

wherein said application service provides execution and management of one or more application programs accessible by said controller using application data stored in said storage to support execution of a multi-user application program,

wherein said message service provides building messages and processing received messages, and

wherein said relay service provides building a new relay message indicating a selected recipient to which the wireless device does not have a direct wireless connection, sending a built new relay message, and sending a received relay message that indicates a recipient other than the wireless device, and

when said selected recipient is a local device, sending said received relay message from said wireless device addressed only to said selected recipient through said wireless communication interface and not sending said received relay message addressed to other local devices,

wherein said controller also supports a recipient selection service, said recipient selection service provides building a list of at least one recipient, and selecting a recipient from said list, and said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service,

wherein said recipient selection service responds to a map request from another device by sending a request to at least one other device,

wherein said map request indicates a hop count, and said recipient selection service does not send said request if said hop count has reached a threshold.

2. The wireless device of claim **1**, wherein:

said recipient selection service provides sending said list to a requesting device.

3. The wireless device of claim **1**, wherein:

said recipient selection service supports building said list using propagation.

4. The wireless device of claim **1**, wherein:

said map request indicates a network identifier, and said recipient selection service does not send said network identifier does not match an identifier stored by the wireless device.

5. The wireless device of claim **1**, wherein:

said recipient selection service periodically updates said list.

6. The wireless device of claim **1**, wherein:

said recipient selection service updates said list using one or more update rules.

7. The wireless device of claim **1**, wherein:

said controller also supports a recipient selection service, said recipient selection service provides building a map of at least one recipient and connections among said at least one recipient and the wireless device, and selecting a recipient from said map, and said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service.

8. The wireless device of claim **7**, wherein:

said built new relay message also indicates at least one intermediary recipient providing an indirect connection between said selected recipient and the wireless device.

9. The wireless device of claim **1**, wherein:

said selected recipient used by said relay service in building a new relay message is selected from a list provided by a server.

10. The wireless device of claim **1**, wherein:

at least one of said one or more application programs executed and managed by said application service is a game application program.

11. The wireless device of claim **1**, wherein:
said wireless interface is a Wi-Fi interface.

12. A network device, comprising:
a network communication interface supporting network communication across a network connection, the network communication interface enabled to contact local devices, which are devices that are within range of a sending device through the network communication interface, and requesting identification of available devices;
storage supporting storing data; and
a controller connected to said network interface and to said storage, supporting an application service, a message service, and a relay service for relay messages, the controller enabled to (1) propagate by the local devices the request for identification of available devices through a peer-to-peer network, (2) receive the identification of available devices to the sending device, and (3) build a list of available devices, the list including identification of said local devices in a local device list;
wherein said application service provides execution and management of one or more application programs accessible by said controller using application data stored in said storage to support execution of a multi-user application program,
wherein said message service provides building messages and processing received messages, and
wherein said relay service provides building a new relay message indicating a selected recipient to which the network device does not have a direct connection, sending a built new relay message, and sending a received relay message that indicates a recipient other than the network device, and
when said selected recipient is a local device, sending said received relay message from said wireless device addressed only to said selected recipient through said network communication interface and not sending said received relay message addressed to other local devices,
wherein said controller also supports a recipient selection service, said recipient selection service provides building a list of at least one recipient, and selecting a recipient from said list, and said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service,
wherein said recipient selection service responds to a map request from another device by sending a request to at least one other device,
wherein said map request indicates a hop count, and said recipient selection service does not send said request if said hop count has reached a threshold.

13. The network device of claim **12**, wherein:
said controller also supports a recipient selection service, said recipient selection service provides building a map of at least one recipient and connections among said at least one recipient and the network device, and selecting a recipient from said map, and said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service.

14. The network device of claim **13**, wherein:
said built new relay message also indicates at least one intermediary recipient providing an indirect connection between said selected recipient and the network device.

15. The network device of claim **12**, wherein:
said selected recipient used by said relay service in building a new relay message is selected from a list provided by a server.

16. The network device of claim **12**, wherein:
at least one of said one or more application programs executed and managed by said application service is a game application program.

17. A wireless game device, comprising:
an antenna;
a wireless communication interface connected to said antenna and supporting wireless communication across a wireless connection provided by said antenna, the wireless communication interface enabled to contact local devices, which are devices that are within range of a sending device through the wireless interface, and requesting identification of available devices;
storage supporting storing data;
a media interface connected to said storage supporting reading a game application program from removable media provided to said media interface; and
a controller connected to said wireless interface, to said storage, and to said media interface, supporting an application service, a message service, a relay service for relay messages, and a recipient selection service, the controller enabled to (1) propagate by the local devices the request for identification of available devices through a peer-to-peer network, (2) receive the identification of available devices to the sending device, and (3) build a list of available devices, the list including identification of said local devices in a local device list;
wherein said application service provides execution and management of one or more game application programs accessible by said controller using application data stored in said storage,
wherein said message service provides building messages and processing received messages, and
wherein said relay service provides building a new relay message indicating a selected recipient to which the wireless game device does not have a direct wireless connection, sending a built new relay message, and sending a received relay message that indicates a recipient other than the wireless game device,
wherein said recipient selection service provides building a map of at least one recipient and connections among said at least one recipient and the wireless game device, and selecting a recipient from said map, said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service, and said wireless interface is a Wi-Fi interface, and
when said selected recipient is a local device, sending said received relay message from said wireless device addressed only to said selected recipient through said wireless communication interface and not sending said received relay message addressed to other local devices,
wherein said controller also supports a recipient selection service, said recipient selection service provides building a list of at least one recipient, and selecting a recipient from said list, and said selected recipient used by said relay service in building a new relay message is a recipient selected by said recipient selection service,
wherein said recipient selection service responds to a map request from another device by sending a request to at least one other device,
wherein said map request indicates a hop count, and said recipient selection service does not send said request if said hop count has reached a threshold.

* * * * *

EXHIBIT 89

US007831666B2

(12) **United States Patent**
Chatani et al.

(10) **Patent No.: US 7,831,666 B2**
(45) **Date of Patent: *Nov. 9, 2010**

(54) **MANAGING PARTICIPANTS IN AN ONLINE SESSION**

(75) Inventors: **Masayuki Chatani**, Foster City, CA (US); **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Foster City, 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.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/403,623**

(22) Filed: **Apr. 13, 2006**

(65) **Prior Publication Data**

US 2006/0190540 A1 Aug. 24, 2006

Related U.S. Application Data

(63) Continuation of application No. 11/375,526, filed on Mar. 13, 2006, which is a continuation of application No. 10/211,128, filed on Jul. 31, 2002.

(60) Provisional application No. 60/381,736, filed on May 17, 2002.

(51) **Int. Cl.**
G06F 15/16 (2006.01)
A63F 9/14 (2006.01)

(52) **U.S. Cl.** **709/205**; 709/209; 463/62

(58) **Field of Classification Search** 709/208, 709/204, 205, 209; 463/42
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,630,757 A 5/1997 Gagin et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1125617 8/2001

(Continued)

OTHER PUBLICATIONS

Festa et al., "Netscape Alumni to Launch P2P Company," Aug. 2, 2001.

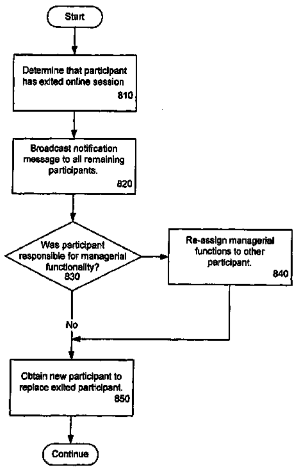
(Continued)

Primary Examiner—Douglas B Blair
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. The application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

21 Claims, 11 Drawing Sheets


Start
Determine that participant has exited online session 810
Broadcast notification message to all remaining participants. 820
Was participant responsible for managerial functionality? 830
Re-assign managerial functions to other participant. 840
No
Obtain new participant to replace exited participant. 850
Continue

U.S. PATENT DOCUMENTS

5,682,139	A *	10/1997	Pradeep et al.	340/539.13
5,704,032	A	12/1997	Badovinatz et al.	
5,838,909	A *	11/1998	Roy et al.	709/209
5,841,980	A	11/1998	Waters et al.	
5,893,106	A	4/1999	Brobst et al.	
5,956,485	A	9/1999	Perlman	
5,984,787	A	11/1999	Redpath	
5,987,376	A	11/1999	Olson et al.	
6,041,312	A	3/2000	Bickerton et al.	
6,152,824	A	11/2000	Rothschild et al.	
6,154,782	A	11/2000	Kawaguchi et al.	
6,219,045	B1	4/2001	Leahy et al.	
6,311,209	B1 *	10/2001	Olson et al.	709/204
6,363,416	B1 *	3/2002	Naeimi et al.	709/209
6,487,678	B1	11/2002	Briskey et al.	
6,560,636	B2 *	5/2003	Cohen et al.	709/203
6,561,811	B2 *	5/2003	Rapoza et al.	434/236
6,607,444	B2	8/2003	Takahashi et al.	
6,631,412	B1 *	10/2003	Glasser et al.	709/224
6,676,521	B1	1/2004	La Mura et al.	
6,748,420	B1	6/2004	Quatrano et al.	
6,761,636	B2 *	7/2004	Chung et al.	463/42
6,931,446	B1	8/2005	Cox et al.	
7,003,550	B1 *	2/2006	Cleasby et al.	709/205
7,016,942	B1	3/2006	Odom	
7,018,295	B2 *	3/2006	Sakaguchi et al.	463/42
7,056,217	B1	6/2006	Pelkey et al.	
7,107,312	B2 *	9/2006	Hackbarth et al.	709/204
7,290,264	B1 *	10/2007	Powers et al.	719/315
7,587,465	B1 *	9/2009	Muchow	709/209
7,720,908	B1 *	5/2010	Newson et al.	709/204
7,730,206	B2 *	6/2010	Newson et al.	709/240
2001/0009868	A1 *	7/2001	Sakaguchi et al.	463/42
2001/0044339	A1 *	11/2001	Cordero et al.	463/42
2002/0035604	A1	3/2002	Cohen et al.	
2002/0049086	A1 *	4/2002	Otsu	463/42
2002/0062348	A1	5/2002	Maehiro	
2002/0075844	A1	6/2002	Hagen	
2002/0133707	A1 *	9/2002	Newcombe	713/183
2003/0018719	A1 *	1/2003	Ruths et al.	709/205
2003/0073494	A1 *	4/2003	Kalpakian et al.	463/42
2003/0217135	A1	11/2003	Chatani	
2003/0217158	A1	11/2003	van Datta	
2004/0117443	A1 *	6/2004	Barsness	709/204
2005/0251577	A1	11/2005	Guo	
2005/0259637	A1	11/2005	Chu	
2006/0100020	A1 *	5/2006	Kasai	463/42
2006/0253595	A1	11/2006	van Datta	
2006/0288103	A1	12/2006	Gobara	
2007/0058792	A1	3/2007	Chaudhari	
2007/0076729	A1	4/2007	Takeda	
2007/0165629	A1	7/2007	Chaturvedi	
2007/0191109	A1	8/2007	Crowder	
2007/0208748	A1	9/2007	Li	
2008/0280686	A1 *	11/2008	Dhupelia et al.	463/42
2009/0077245	A1	3/2009	Smelyansky	
2009/0138610	A1	5/2009	Gobara	
2009/0240821	A1	9/2009	Juncker	

FOREIGN PATENT DOCUMENTS

WO	WO 00/05854	2/2000
WO	WO 01/82678	11/2001
WO	WO 02/35769	5/2002

OTHER PUBLICATIONS

Diot et al, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet," IEEE Vol. 13, Issue 4, Aug. 1999.
European Search Report for Ep 03 72 1413, Jun. 30, 2005.
Cisco Systems, Inc., "Network Flow Management: Resource Reservation for Multimedia Flows," Mar. 19, 1999. - .
"Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Mar. 23, 2007).
"Brief for Appellee," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (May 21, 2007).
"Reply Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Jun. 4, 2007).
"Petition for Panel Rehearing," In Re Masayuki Chatani and Glen Van Datta, Appeal From the United States Patent and Trademark Office, Boar of Patent Appeals and Interferences, In the United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 10/211,128), Jan. 3, 2008.
"In Re Masayuki Chatani and Glen Van Datta," United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 11/211,128), Nov. 19, 2007.
Hagsand O; "Interactive Multiuser VEs in the DIVE System" IEEE Multimedia, IEEE Service Center, New York, NY, US vol. 3, No. 1, Mar. 21, 1996, pp. 30-39, XP000582951 ISSN:1070-986X.

* cited by examiner

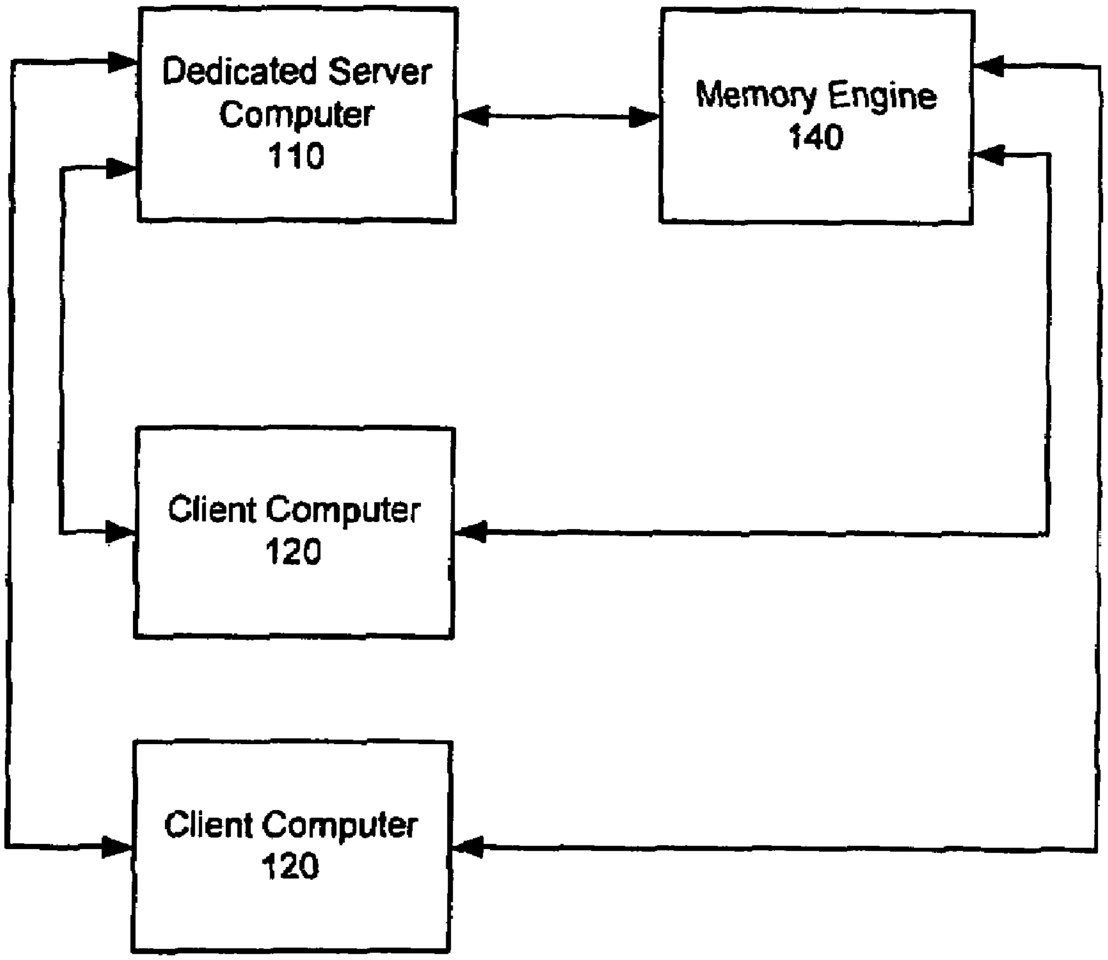


Figure 1

PRIOR ART

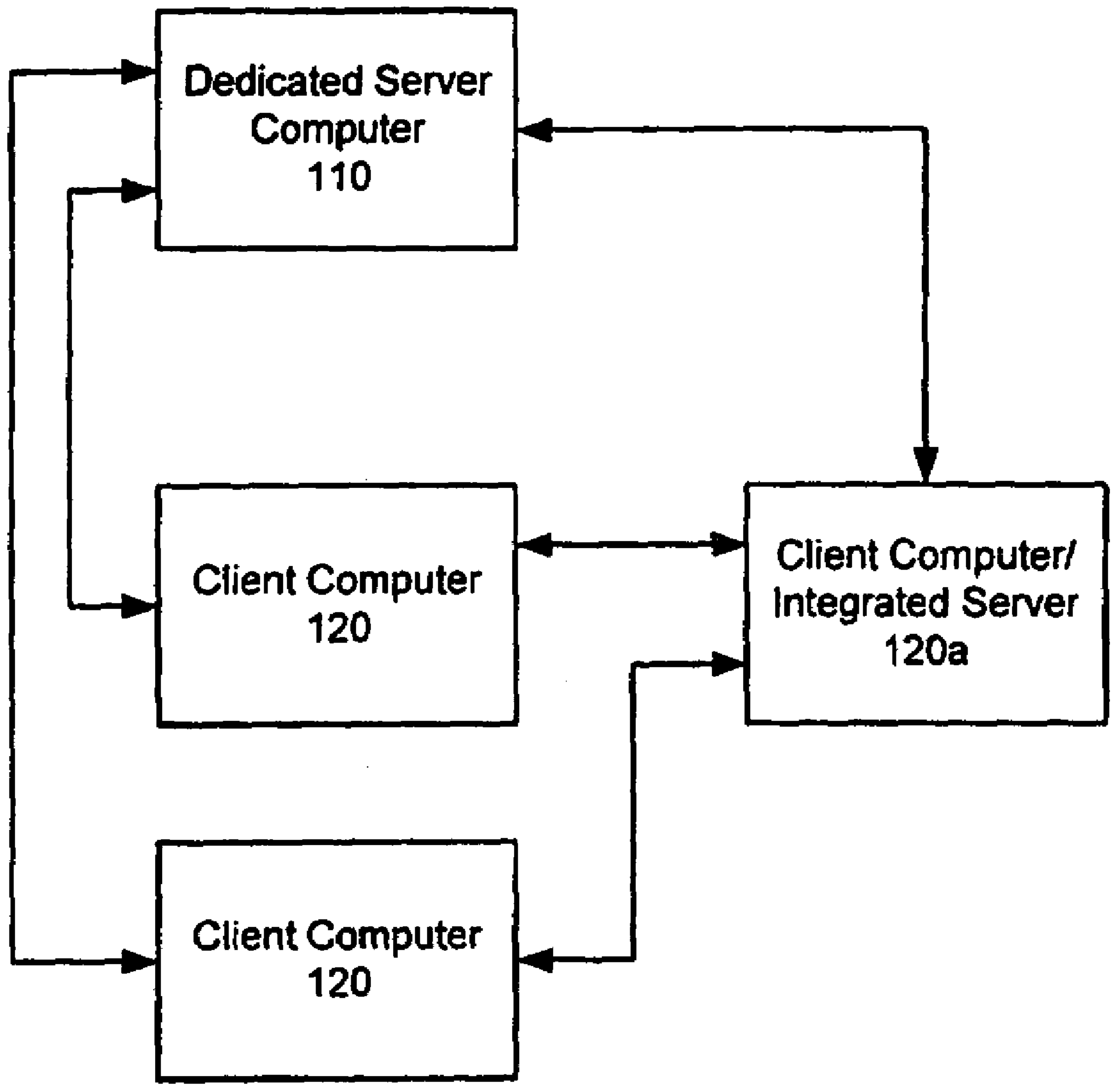

Dedicated Server Computer 110
Client Computer 120
Client Computer/ Integrated Server 120a
Client Computer 120

Figure 2

PRIOR ART

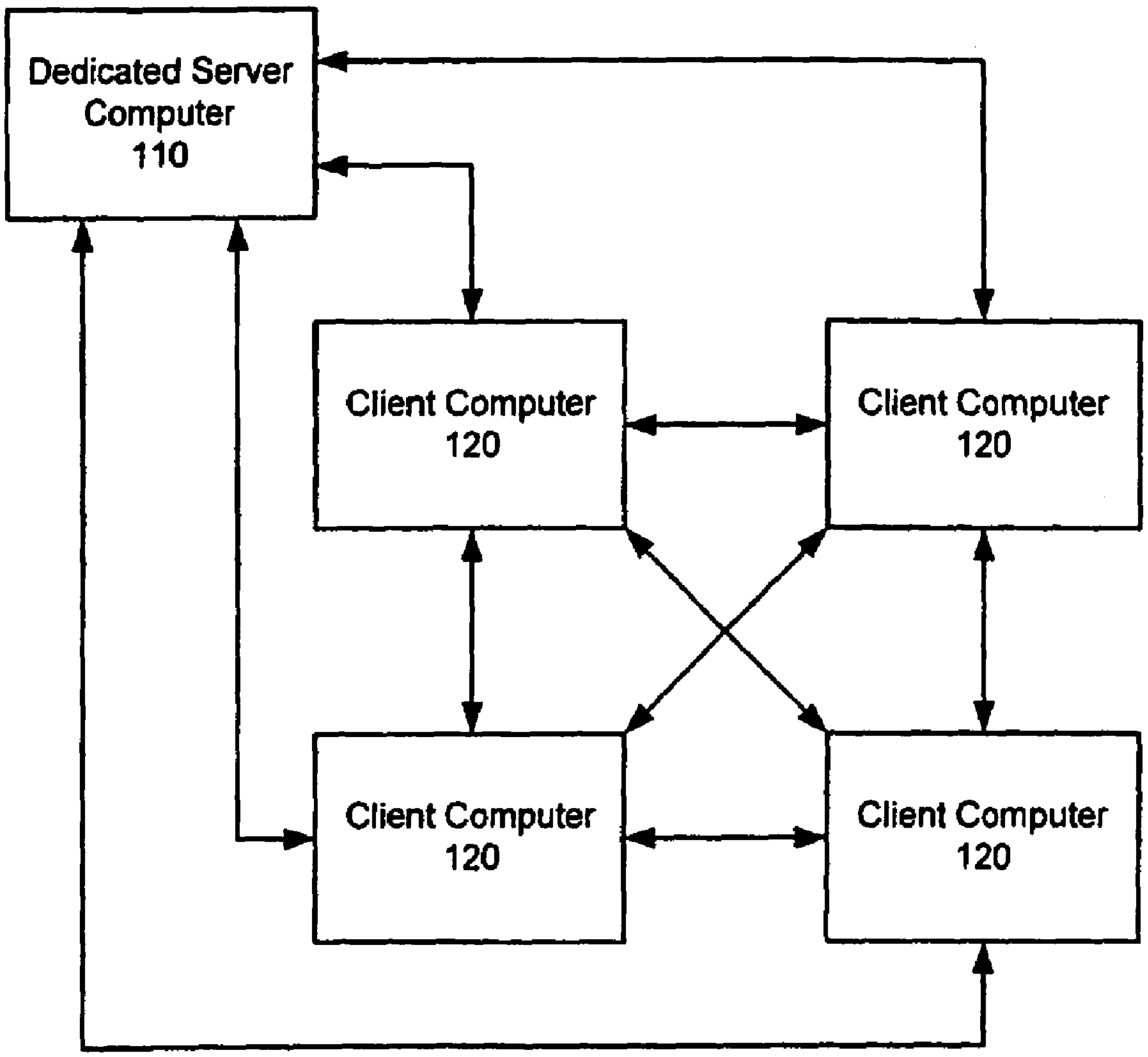


Figure 3

PRIOR ART

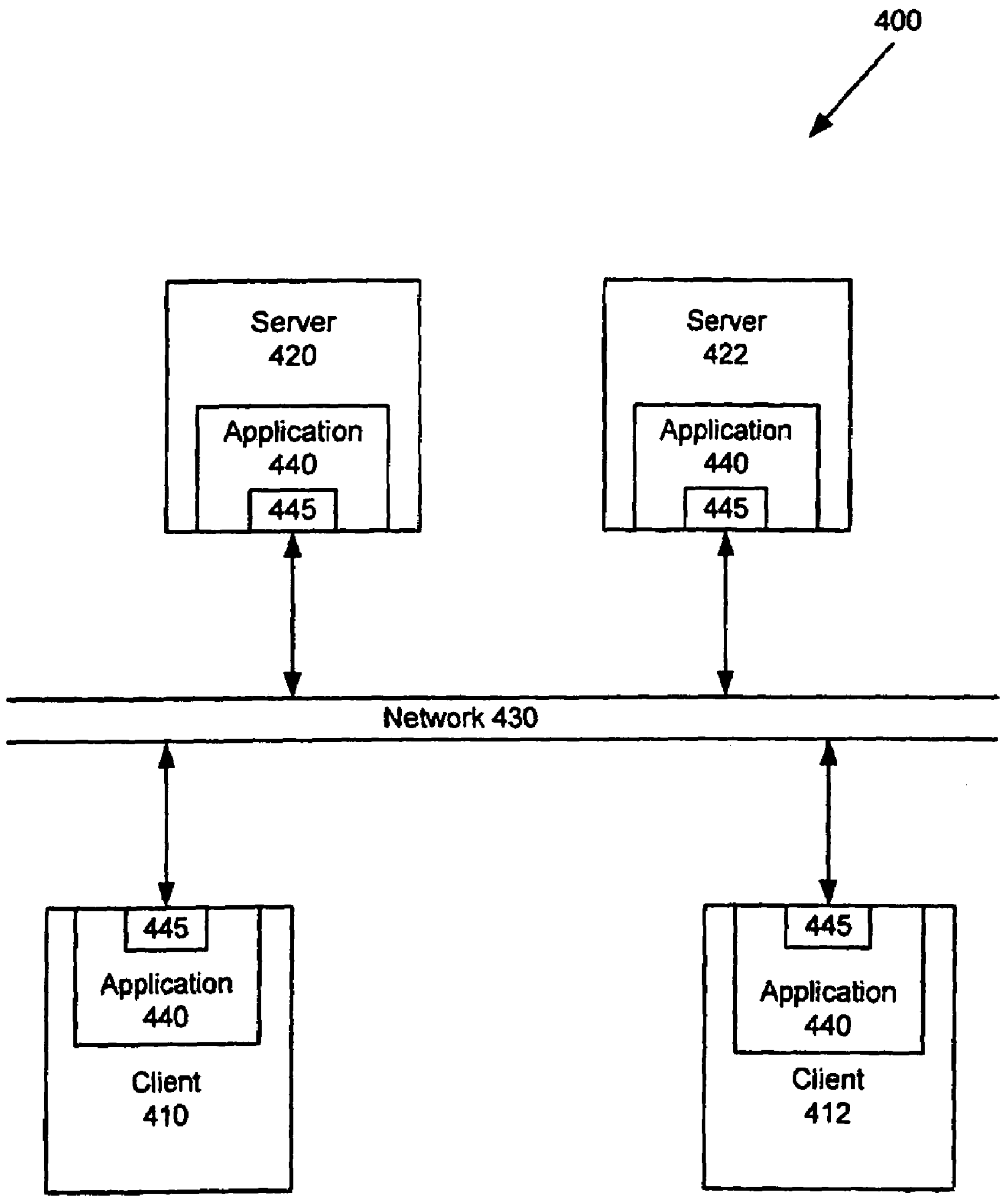

400
Server 420
Application 440
445
Server 422
Application 440
445
Network 430
445
Application 440
Client 410
445
Application 440
Client 412

Figure 4

500

Index	Session Master	Active Port/Protocol Type
1	Owns C1	80/TCP, 90/UDP
2	Owns C2, C3	85/TCP, 95/UDP
3	-	
-	-	
-	-	
-	-	

Figure 5

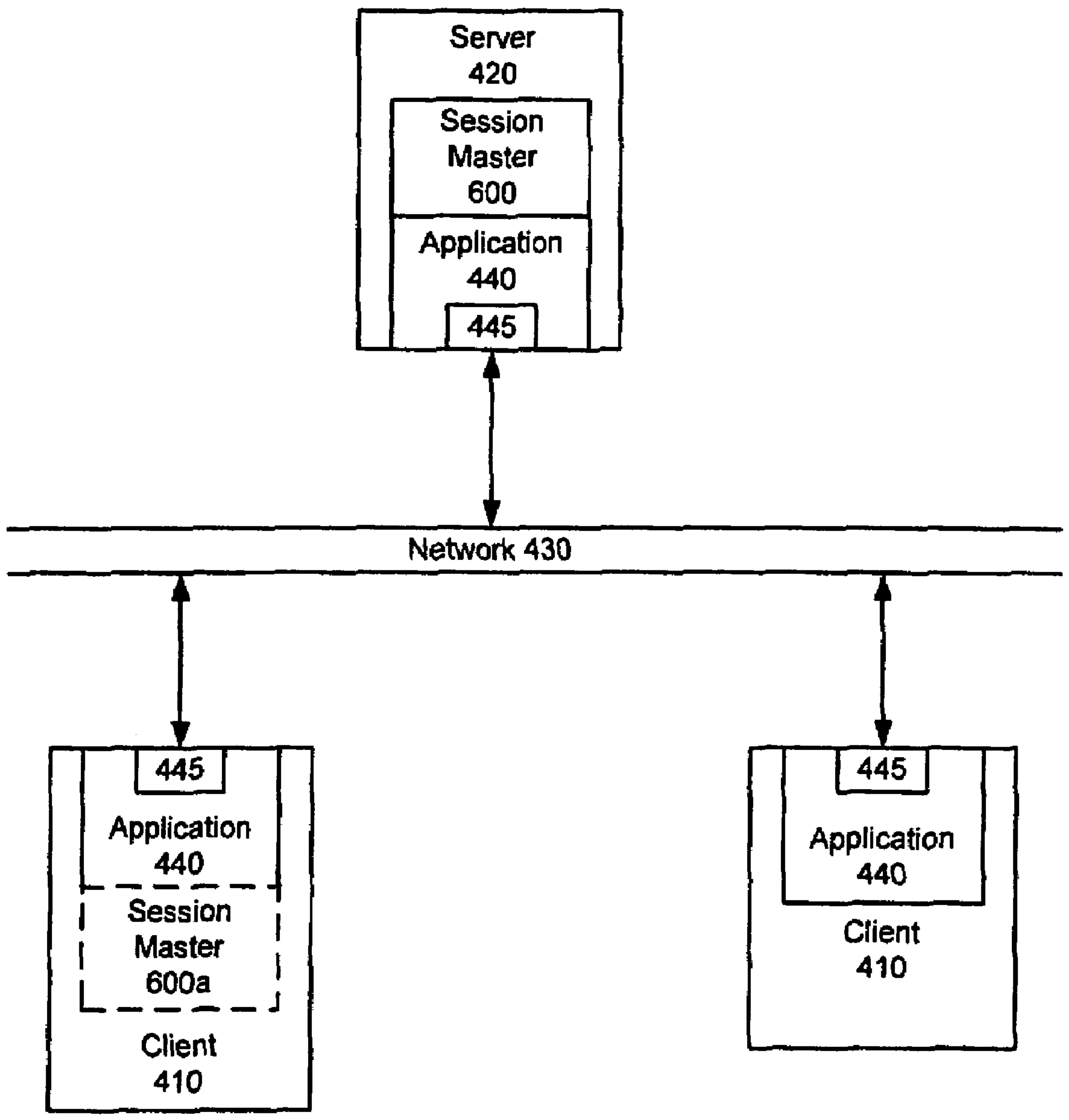

Server
420
Session
Master
600
Application
440
445
Network 430
445
Application
440
Session
Master
600a
Client
410
445
Application
440
Client
410

Figure 6

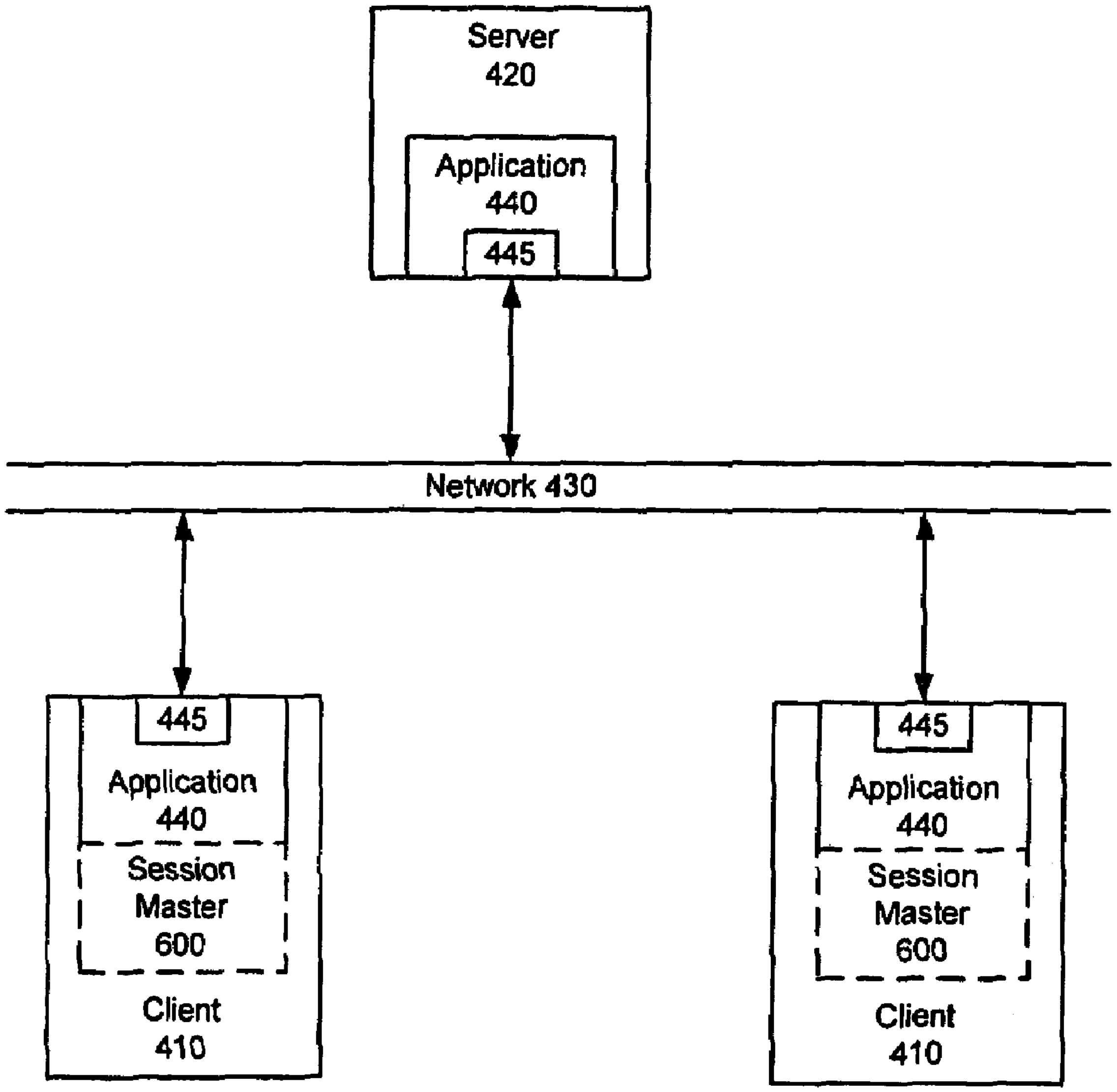

Server
420
Application
440
445
Network 430
445
Application
440
Session
Master
600
Client
410
445
Application
440
Session
Master
600
Client
410

Figure 7

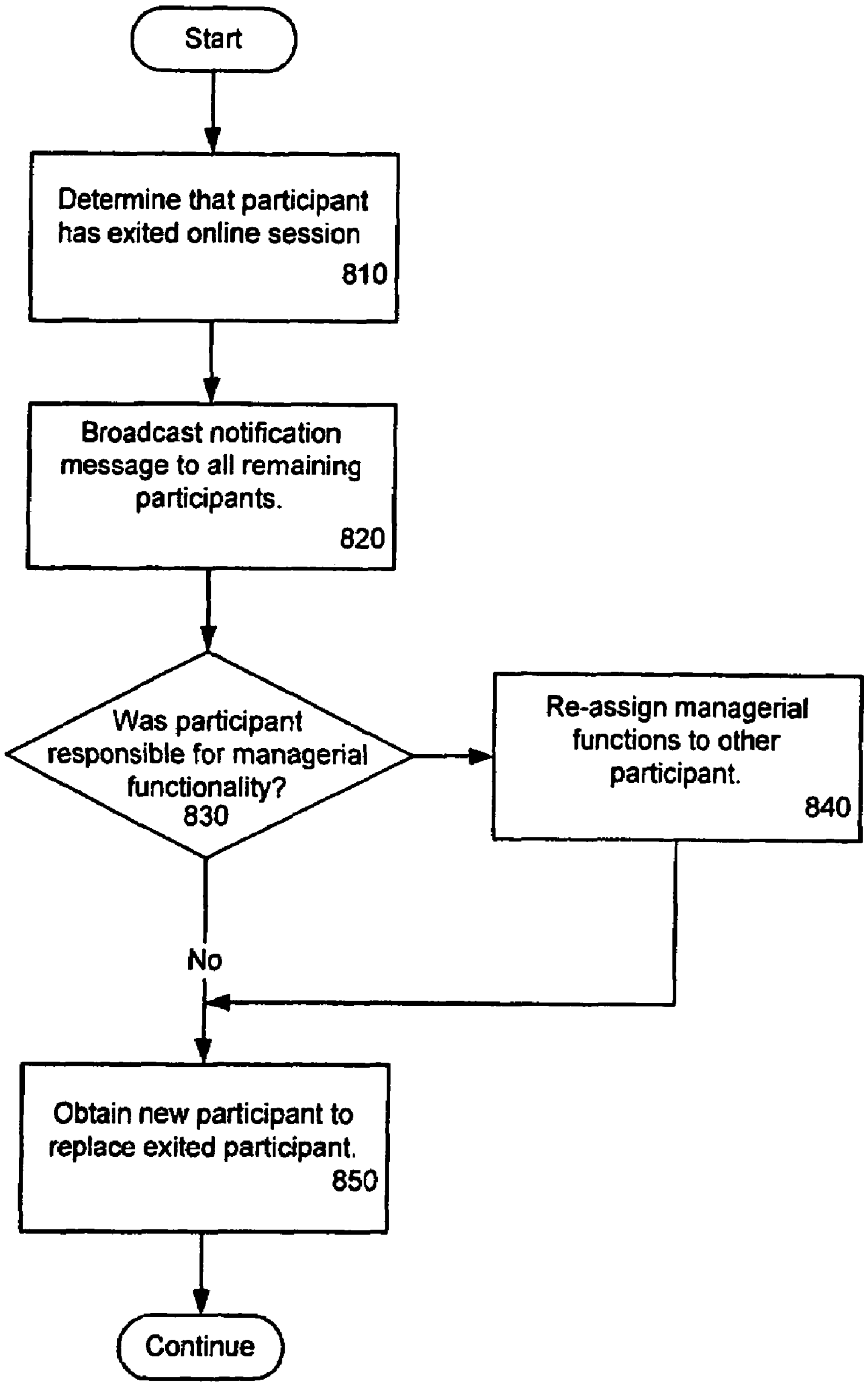

Start
Determine that participant has exited online session
810
Broadcast notification message to all remaining participants.
820
Was participant responsible for managerial functionality?
830
Re-assign managerial functions to other participant.
840
No
Obtain new participant to replace exited participant.
850
Continue

Figure 8

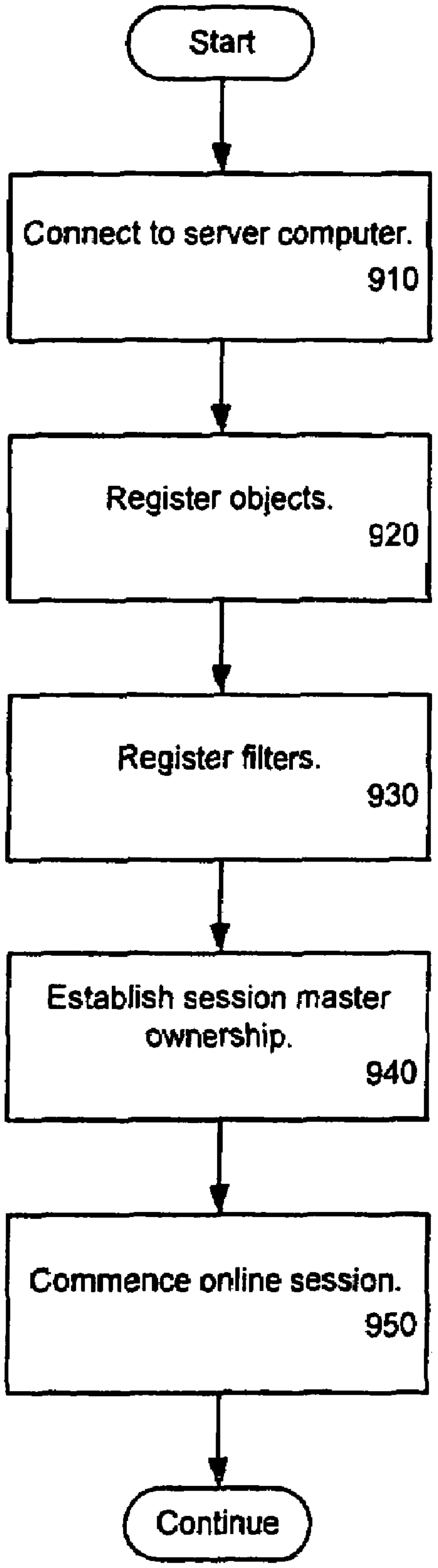

Start
Connect to server computer.
910
Register objects.
920
Register filters.
930
Establish session master ownership.
940
Commence online session.
950
Continue

Figure 9

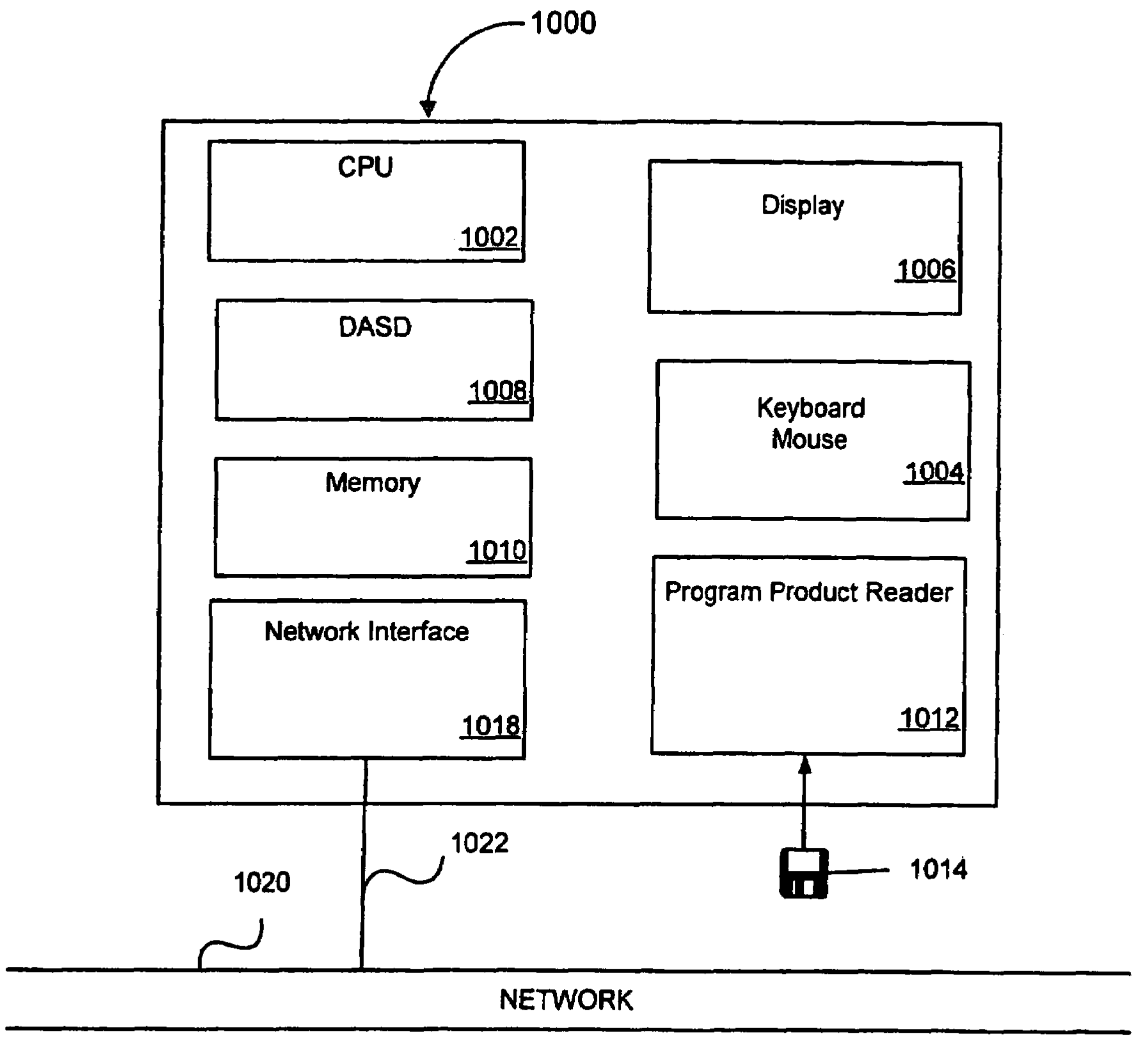

1000
CPU
1002
DASD
1008
Memory
1010
Network Interface
1018
Display
1006
Keyboard
Mouse
1004
Program Product Reader
1012
1022
1020
1014
NETWORK

Figure 10

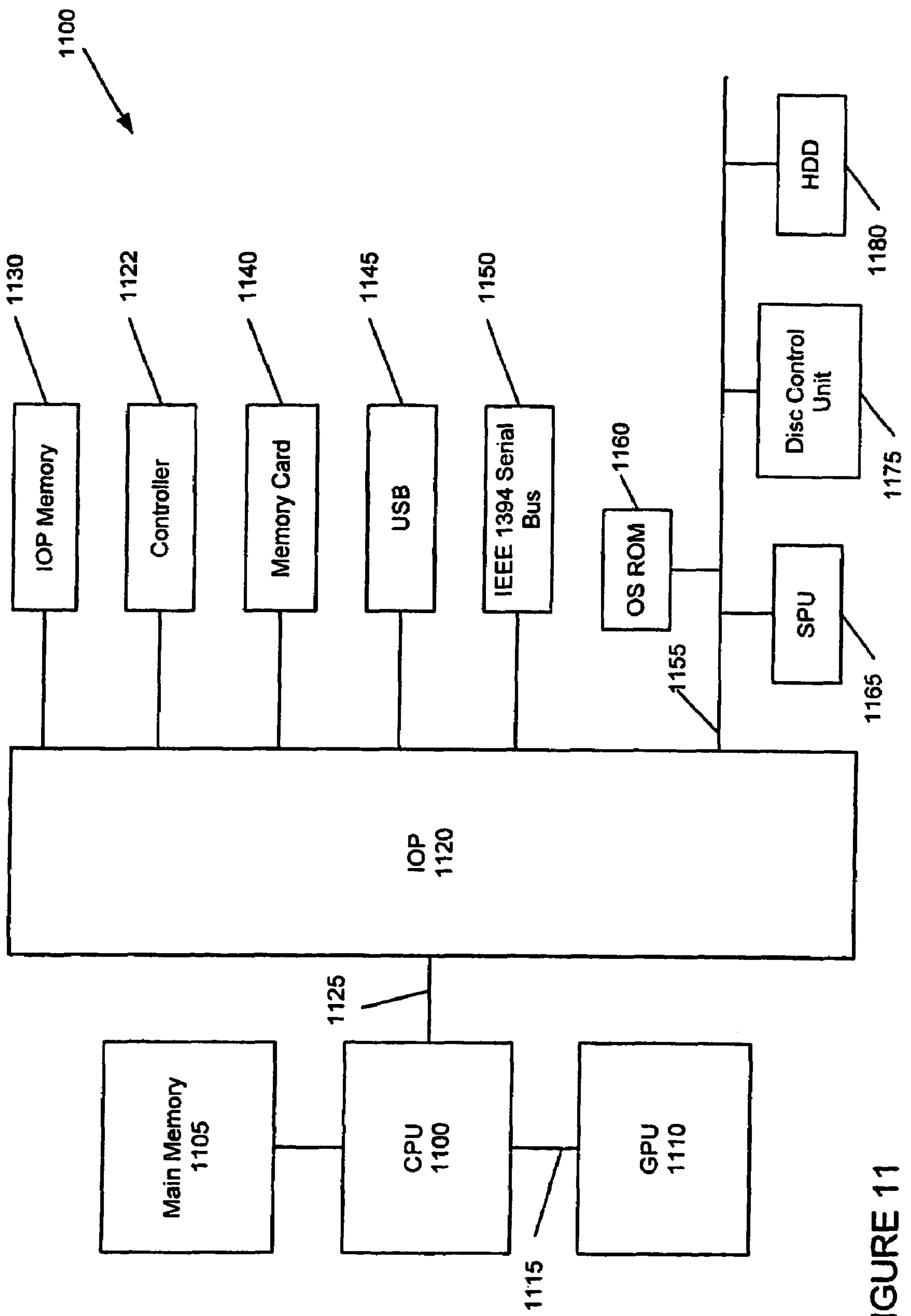

1100
IOP Memory
1130
Controller
1122
Memory Card
1140
USB
1145
IEEE 1394 Serial Bus
1150
OS ROM
1160
1155
SPU
1165
Disc Control Unit
1175
HDD
1180
IOP
1120
1125
Main Memory
1105
CPU
1100
GPU
1110
1115

FIGURE 11

MANAGING PARTICIPANTS IN AN ONLINE SESSION

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation and claims the priority benefit of U.S. patent application Ser. No. 11/375,526 filed Mar. 13, 2006 and entitled "Managing Participants in an Online Session," which is a continuation and claims the priority benefit of U.S. patent application Ser. No. 10/211,128 filed Jul. 31, 2002 and entitled "Dynamic Player Management," which claims the priority benefit of U.S. provisional patent application No. 60/381,736 filed May 17, 2002 and entitled "Dynamic Player Management." The disclosure of these commonly owned applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to computer networks and, more particularly, to an application that is run by multiple users on a computer network.

2. Description of the Related Art

Computer networks, such as local area networks and the Internet, are increasingly being used as the backbone for various transactions and interactions between parties. From online banking, where bank customers can initiate financial transactions on a computer network, to online gaming, where garners can participate in various games over the Internet, service providers are increasingly providing a variety of services over computer networks. There are currently a variety of different computer network configurations that facilitate the transactions and interactions that take place.

One type of configuration is a classic client-server configuration, such as is illustrated in FIG. **1**. In this configuration, a dedicated server computer **110** is communicatively linked to one or more client computers **120** over a network, such as through the Internet. The client computer **120** makes service requests to the server computer **110** and the server computer **110** fulfills the request by transmitting data to the requesting client computer **120** over the network. The server computer **110** can be connected to a data storage device or to other computer devices that facilitate transactions between the client and server computers. One characteristic of the client-server configuration is that the client computers cannot communicate directly with one another, as the client computers are limited to communicating with the server computer.

For example, where the client-server configuration is operated in an online gaming environment, the server computer **110** can be responsible for maintaining the various states that are associated with the online game. The server computer can be connected to other computers, such as a memory engine **140** that maintains one or more instances of a game, while the server computer **110** manages administrative matters such as player matching and account management. A game player on the client computer **120** can log onto the server computer **110** and receive a list of available games and participating players. The player chooses a game to start or join, thereby identifying a memory engine with which the player's computer establishes a client-server connection. In this manner, the server computer **110** and the memory engine **140** collectively administer the gaming environment for one or more client computers **120**.

Another type of configuration is referred to as an integrated server configuration, such as is shown in FIG. **2**. This configuration includes a dedicated server computer **110** and one or more client computers **120** that are each connected to the server computer **110** over a computer network. As in the previously-described configuration, the server computer **110** serves data to the client computers **120**. However, one of the client computers **120**, such as the client computer **120***a*, functions as an integrated server in that the client computer **120***a* can serve data to the other client computers **120**. In an online gaming environment, the server computer **110** can perform administrative functions, such as player matching, account management, and chat room management, while the client computer/integrated server **120***a* can perform the function of the previously-described memory engine.

In yet another type of communication configuration, the various computers are arranged in a peer-to-peer configuration, such as is shown in FIG. **3**. In a peer-to-peer configuration, each of the computers can communicate with the others, so that all of the computers function as "peers." In one form of the peer-to-peer configuration, a dedicated server **110** is communicatively connected to a plurality of client computers **120** over a network. An online session is initially established by each of the client computers **120** connecting to an administrative computer, such as the server computer **110**. The client computers **120** are then communicatively connected to one another so that each of the client computers **120** has the ability to both serve and receive data to and from any of the other client computers **120**. In addition, each client computer **120** can operate in a client-server relationship with the dedicated server **110**. Those skilled in the art will appreciate that there are other communication configurations in addition to the configurations described above.

The various configurations described above enable computer users to interact over a computer network, such as in an online game environment where game players can play computer games on a computer network. In such a scenario, at least one of the computers typically functions as a game manager that manages various aspects of the game, such as coordinating the number of players, keeping track of game state, and sending out updates to the users regarding game state. It can be appreciated that continuity of game play can be highly dependent on all of the users in a game continuing to play throughout the entire game period. Game play can be interrupted or even halted if one of the game players exits during the middle of a game, particularly where the exited player was managing a portion of the game.

For example, sports games typically have a fixed start and a fixed finish for the game, with at least two players competing in a game. In current configurations, there are often several players that participate in an online sporting contest, with each one of the players assuming the role of a player on a sporting team. For example, in an online football game, players can assume the roles of quarterback, receiver, defensive back, running back, etc. If one of the players were to suddenly leave during the middle of the game, then the game play would be interrupted or halted. This could also be the case in other types of games, where continuity of game play is dependent on each of the players in the game environment continuing to play throughout a particular scenario.

Unfortunately, current multi-user applications are not configured to account for when a participant in an online session suddenly or unexpectedly leaves the online session. If a player does leave an online session, then the session is unduly interrupted or terminated. In view of the foregoing, there is a need for a multi-user application that overcomes the aforementioned shortcomings.

SUMMARY OF THE INVENTION

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. In accordance with one aspect of the invention, the application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

Other features and advantages of the present invention should be apparent from the following description of the preferred embodiment, which illustrates, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of a computer network arranged in a client-server network communication configuration.

FIG. **2** is an illustration of a computer network arranged in an integrated network communication server configuration.

FIG. **3** is an illustration of a computer network arranged in a peer-to-peer network communication configuration.

FIG. **4** is an illustration of a computer network system on which is run a multi-user application configured in accordance with the present invention.

FIG. **5** is an illustration of a data structure that includes computer index and session master information for the multi-user application.

FIG. **6** is an illustration of a computer network system wherein the multi-user application is arranged in a first type of communication configuration.

FIG. **7** is an illustration of a computer network system wherein the multi-user application is arranged in another type of communication configuration.

FIG. **8** is a flow diagram that represents a process of managing the exiting of a participant in an online session of a multi-user application.

FIG. **9** is a flow diagram that illustrates the operating steps associated with the multi-user application establishing an online session.

FIG. **10** is a block diagram of a computer in the network illustrated in FIG. **4**, illustrating the hardware components.

FIG. **11** is a block diagram of a computer entertainment system in the network illustrated in FIG. **4**, illustrating the hardware components.

DETAILED DESCRIPTION

FIG. **4** is a block diagram of a computer network system **400** comprised of one or more network devices including one or more client computers **410**, **412** and one or more dedicated server computers **420**, **422**, which are nodes of a computer network **430**. Thus, some of the network computers are configured as servers and some are configured as clients. The computer network **430** may comprise a collection of interconnected networks, such as the Internet, and may include one or more local area networks at each of the nodes **410**, **412**, **420**, **422**. As used herein, the term "Internet" refers to a collection of interconnected (public and/or private) networks that are linked together by a set of standard communication protocols to form a global, distributed network.

The client computers **410**, **412** can transmit requests for data over the network **430** to one of the server computers **420**, **422**, which are configured to serve data over the network **430** to the client computers in a well-known manner. The server computers **420**, **422** can include or can be communicatively linked to each other and to other servers, such as a data base server and/or an application server, as will be known to those skilled in the art. Although FIG. **4** shows only two client computers **410**, **412** and two server computers **420**, **422**, it should be appreciated that the network system **400** could include any number of client computers **410**, **412** and server computers **420**, **422**. The server computers **420**, **422** and client computers **410**, **412** are sometimes referred to collectively herein as network computers.

The network system **400** supports a multi-user application **440** comprised of a computer program with which multiple users can interact in online sessions using network devices (such as the client computers **410**, **412**) that are linked to the computer network **430**. The application **440** is installed at each of the client computers, meaning that an operational instance of the application is stored in memory of each of the client computers **410**, **412** that run (execute) the application **440**. Each server computer that will be participating in an online session of the multi-user application also stores an instance of the application **440**. For purposes of this description, the first server computer **420** will be assumed to be a server for the multi-user application being executed by the client machines **410**, **412**, although both servers **420**, **422** are shown with installed applications **440**. An exchange of data occurs between instances of the application **440** during execution and is enabled through the establishment of network sockets **445** at each of the network computers. The sockets are represented in FIG. **4** as boxes at each respective network computer. Those skilled in the art will understand that a network socket is one end of a multi-way communication link between two or more programs that run on the network system **400**.

The application **440** can be run on the network devices of the network system **400** according to a variety of communication configurations and the responsibilities for various application-related processes can be assigned to different computing devices of the network **430**, as described in more detail below. An application development interface is preferably used to develop the application **440**, as is also described in more detail below. The application can be operated such that the associated network computer can use a communications configuration to implement any of the communication modes illustrated in FIG. **1**, FIG. **2**, and FIG. **3**.

The multi-user application **440** can be any type of application that a user can run on a network computer that is linked to the computer network **430**. When the application **440** is run on a client computer **410**, **412**, the user can interact with other users through other network computers that are also running the application **440**. The server computer **420** can function as a central network "meeting point" through which the users can establish contact, maintain data, and initiate an online session of the application **440**. Typically, the application **440** causes the network device in which it is operating to establish communications with another network device, such as the devices **410**, **412**, **420**, thereby initiating an online session. During the online session, the network computers will interact and exchange data pursuant to the programmed features of the application **440**.

When the application **440** is launched and an online session is established among suitably configured computers, the application enables the computers to interact in a variety of configurations. Throughout this description, the application **440** is sometimes described in an online-gaming scenario, wherein the application **440** comprises a computer game that multiple users can access and run using the client computers **410**, **412**. In such a case, the application **440** establishes an online session comprised of a game in which the network computers participate. However, it should be appreciated that the application **440** may also relate to other scenarios besides gaming, such as, for example, online banking or online travel planning, that involve interactions between multiple computers on a computer network.

When an application **440** executes, it identifies a session master, which is a network computer that performs a variety of managerial and administrative functions for the application with respect to interactions between computers that occur during an online session. An online session of the application uses a registration or logon process with a data store containing information such as user identification. The logon process authorizes further participation in the network environment of the application. Preferably, the session master function is assigned when a client computer running the application **440**, such as the client computer **410**, logs onto the server computer **420** to initiate an online session. The application itself, however, determines the details of when and how such assignments are made, so that a variety of session master assignment schemes can be implemented without departing from the teachings of the present invention.

The operating instance of the application on the client computer **410** that initiates an online session of the application is referred to as the host computer. The application at the host computer assigns the session master function to either the server computer **420** or to the host computer **410**. As new client computers log-on (register) with the server computer **420** to join the online session, the server computer **420** notifies the new clients of the already-assigned identity of the session master computer.

As described more fully below, the session master functionality enables a smooth transition between the various network communication configurations in which the application **440** can operate. The session master function also enables the application **440** to concentrate responsibility for application-related tasks in a specific network computer, or to distribute such responsibility among two or more network computers. The assignment of tasks can be performed by an instance of the application **440** on one of the network computers, at the same time when the session master function is assigned, and the session master tasks can be assigned to one or more of the computers on the network **430** for providing the requisite functionality. The computer or computers that are assigned responsibility of the session master are referred to herein as the "owners' of the respective session master functions. References to a solitary session master will be understood to apply to a group of computers, if those computers are collectively performing the session master functions. Thus, the assignment of session master tasks is performed in the manner specified by the application, in accordance with the dictates of the application developer.

One category of responsibilities assigned to the session master relates to application-specific functions, which are functions that are peculiar to the particular type of application **440** being executed. For example, if the application **440** is a game-type application, then the session master or a group of session masters can keep track of game-type data, such as the game score and the time remaining in the game, and can perform game functions, such as terminating the online session when a game ends. A session master computer can also be assigned the responsibility of keeping track of specific game data, such as the state of an object in the game environment, such as a football, aircraft, ocean, tree, and so forth. Each of these responsibilities can be concentrated in a single session master computer or can be divided up among several session master computers, in accordance with the operation of the application.

The host computer performs managerial functions related to the computers that are participating in the online session. For example, whenever a network computer joins an online session of the application **440**, the host computer assigns an identification index number to the computer joining the session. The host computer maintains a list of the identification index numbers and their associated network computers. The index number is used when sending messages and is also used to maintain ownership records regarding the session master functionality.

As noted above, there can be more than one session master in an online session. The way in which a session master is assigned can be determined by the application in accordance with the operation of the application. The application **440** can also assign the session master with responsibility for sending out update messages to update the network computers regarding the status of all network computers that are participating in the online session. This responsibility entails the session master notifying the participating network computers when a new network computer joins the online session, or when a current participant exits the online session of the application **440**, as described more fully below.

The host computer that assigns the aforementioned index number to each of the computers also maintains a list of all network computers that are participating in the online session. The application **440** then keeps track of session master ownership according to the index number assigned to the computer. To keep track of the index number and responsibility assignments, the application **440** can maintain a data structure such as in the form of a table comprised of a network computer index list, such as the table **500** shown in FIG. **5**. The table **500** contains an index number associated with each network computer that is participating in the online session, and also contains an indication of whether the network computer owns the session master function. The index list data structure comprising the table **500** preferably also specifies the communication protocol that is being used for each network computer. FIG. **5** shows that different session master tasks (C**1**, C**2**, C**3**) can be owned by different network computers.

In addition to specifying the communication protocol, the data structure also specifies, for each network computer, the port for which the communication protocol is associated. Each instance of the application **440** enables the associated network computer that is participating in the online session to open multiple communication ports, with each port being associated with a particular communication protocol. The network computers communicate with other network computers using a particular port and a particular protocol, which is specified in the data structure comprised of the table **500** shown in FIG. **5**. The ports can comprise network sockets through which the instances of the application **440** communicate over the network. The network computers preferably communicate the port/protocol information, as well as the other information contained in the index list, by periodically sending communication messages to one another over the network.

Preferably, all of the computers that are participating in the online session keep their own copy of the table **500** index list. It should be appreciated that the table **500** is merely exemplary and that the initiating host application **440** could keep track of client index numbers and session master ownership in other manners using a wide variety of data structure formats. Alternatively, the session computers can share one or more copies of the table.

The initiating application **440** can operate in various communication configurations depending on how the application **440** assigns ownership of the session master. In a first configuration, shown in FIG. **6**, the initiating application **440** has assigned ownership of a session master **600** to a single computer, such as the dedicated server computer **420**. Therefore, this computer **420** has responsibility for all of the tasks associated with the session master function as dictated by the initiating application **440**. Thus, the application **440** operates in a client-server communication configuration with respect to the functions of the session master, with the server computer **420** serving data to the client computers **410** relating to the session master responsibilities.

It should be appreciated that any of the computers that are participating in the online session of the application **440** could have ownership of one or more of the session master tasks, such as where one of the client computers **410** is shown owning a session master **600***a*, which is shown in FIG. **6** using phantom lines. This indicates that the client computer has been assigned and is performing one or more session master tasks, along with or in place of the nominal session master server computer **420**. That is, there can be several instances of a session master among the computers participating in an online session, with each instance of a session master being assigned specific responsibilities and each session master task being assigned to a different network computer, or multiple tasks being assigned to the same computer. For example, FIG. **6** shows a situation where there are two session masters **600** and **600***a*, each being assigned responsibility for certain functionality relating to the online session of the application **440**. The server computer **420** has certain responsibilities and the client computer **410** also has certain responsibilities, as determined by the application. This is an integrated server configuration, where the client computer **410** that owns the session master **600***a* is functioning as an integrated server. An "integrated server" refers to a situation in which all clients send information to a client that is designated as an integrated server, and where that integrated server propagates the information to the other clients. The client acting as the integrated server can also own one or more session master tasks.

In another scheme, shown in FIG. **7**, the application **440** has distributed ownership of the session master **600** among several computers. In the illustrated example, both of the client computers **410** share ownership of the session master **600**. In such a case, both of the computers could perform the functions associated with the session master so that the network computers in FIG. **7** are in a peer-to-peer configuration. Regardless of the particular communication configuration in which the application **440** operates, the online session of the application includes various network participants comprised of the network computers that are running the application and interacting pursuant to the online session. If one of the participants in the online session were to exit the session, then this may result in an interruption for the other participants that remain in the session. In accordance with one aspect of the invention, the application **440** is configured to handle situations where a participant exits an online session in order to minimize the interruption for remaining participants.

This is described in more detail with reference to the flow diagram shown in FIG. **8**, which describes a process of managing the exiting of a participant in an online session of the multi-user application **440**. In the first operation, represented by the flow diagram box numbered **810**, it is determined that a participant of the online session of the application has exited the online session. The determination that a participant has exited the online session can be made in a variety of manners. In one embodiment, the instance of the application **440** on a network computer participating in the online session periodically causes the network computer to broadcast an update message notifying the other network computers of its presence in the online session. If no update message is received from a particular network computer within a predetermined amount of time, then it is deemed that the network computer has exited the online session. During establishment of the online session, the instance of the application **440** on the computer that starts the online session may assign a particular computer, such as the session master computer, with the responsibility for making the determination that a participant has exited the online session.

In the next operation, represented by the flow diagram box numbered **820**, the instance of the application **440** in one of the computers causes a notification message to be broadcast to all of the participants in the online session notifying them that a participant (the "exited participant") has exited the session. Only one of the network computers, such as, for example, the session master computer, broadcasts the notification message to all of the participants. If the session master computer is the exited computer, then one of the other network computers broadcasts the message, such as the computer with the next consecutive index after the session master computer. The notification message preferably includes the index that was previously assigned to the network computer for the exited participant. In this manner, the other participants can identify the exited participant by consulting the index table that was discussed above with respect to FIG. **5**.

The next operation differs based upon whether the participant that exited the session was responsible for performing any session managerial functions that would affect the other participants of the online session, as represented by the decision box numbered **830**. The managerial functions include functions such as filtering communication messages, assigning identification indices, score-keeping, keeping track of session times, keeping track of items that are located in an online world, keeping track of participant locations in an online world, etc. If the participant that exited the session was responsible for performing any such functions, a "Yes" result from the decision box **830**, then the process proceeds to the operation represented by the flow diagram box numbered **840**. In this operation, the application **440** reassigns the managerial responsibilities of the exited participant to a network computer of another participant that is still present in the online session. The reassignment of managerial responsibilities is preferably performed by an instance of the application **440** on a specific computer, such as the computer that has the next consecutive index after the index of the computer that exited the online session. For example, a computer with a first index may exit the session. The instance of the application on computer with the next consecutive index (as specified in the table **500** shown in FIG. **5**) then performs the re-assignment of the exited computer's duties.

The manner in which the application **440** re-assigns the responsibilities can vary. In one embodiment, the application **440** automatically selects the participant that is re-assigned the responsibilities based on certain factors, some of which are related to the conditions of the network computers of the participants. The conditions can include, for example, the communication environment, geographical location, and hardware specification of the network computers, as well as user-specified preferences.

The communication environment relates to whether the network computer of the participant has large bandwidth capabilities, such as through a cable-modem or DSL. Preferably, those participants with higher bandwidth capabilities are given higher preference for assuming the responsibilities of the exited participant. The geographic location of the participants can also be a factor in deciding which participant is re-assigned the responsibilities of the exited participant. For example, a participant that is centrally located to the other participants may be given a higher priority in order to minimize the latency for communications. The hardware specifications of the network computers that are participating in the online session are also a factor. The application **440** may give higher priority to network computers that have hardware capabilities that are most highly suited for the responsibilities that are being re-assigned, such as computers with powerful data processing capabilities.

In another embodiment of the operations of flow diagram box **840**, the application **440** simply randomly re-assigns the responsibilities of the exited participant to another of the participants of the online session. The application **440** can also consider user-specified preferences. Some users may specify to the application **440** that they do not want to be assigned the responsibility of managing any application functionality. The users may also specify that a particular participant should be re-assigned responsibilities should another participant exit the online session. Alternately, the application **440** may cause a message to be broadcast to all of the participants of the online session asking whether any of the participants would like to take over the responsibilities that were previously assigned to the exited participant.

The next operation is represented by the flow diagram box numbered **850**. This operation occurs after the application **440** has re-assigned the responsibilities of the exited participant. The operation of flow diagram box **850** also occurs if the exited participant did not have any responsibilities that needed re-assignment, which would have resulted in a "No" outcome from the decision box numbered **830**. In this operation, the application **440** attempts to obtain a new participant to replace the exited participant. The attempt is preferably performed by the instance of the application **440** on a specific computer, such as the session master computer. It should be appreciated that this operation differs from the operation of flow diagram box **840** in that this operation relates to obtaining a replacement participant for the online session to take the place of the exited participant, rather than re-assigning the managerial functions of the exited participant.

For example, the online session may be an online football game, where the participants are each a player on a common team. One of the participants may have been assigned managerial functions comprised of keeping track of the score and of the game time. That same participant may have played the role of the quarterback in the game. If the participant exits the online game during the game, then the application **440** re-

assigns the managerial functions (i.e., score keeping and game time responsibilities) of the exited participant to another participant in the operation **840** and then, in operation **850**, attempts to obtain a new participant to replace the exited participant's role as quarterback.

The manner in which the application **440** attempts to obtain a replacement participant for the online session may vary. In one embodiment, the application **440** automatically assigns a network computer, rather than a human, as a replacement for the exited participant. The network computer thereby would perform the functions of the exited participant. In another embodiment, the application **440** maintains a list of network computer that might be able to participate in the online session and then sends a message to those computers inviting them to participate in the session. The application **440** may put the online session in a pause mode while a replacement participant is obtained.

The application **440** is preferably developed using a software development kit (SDK) that provides a library of object and communication message definitions that are used in the application **440**. The software development kit includes an application interface through which applications that are developed using the SDK can run on a network system, such as the network system **400**. The application interface can reside in a central network server, such as the server **420**, to which network computers that have the application **440** can log onto in order to operate an online session of the application. By using the object and message types provided by the SDK, the application **440** can be developed to include the features described above. The SDK preferably includes an object definition structure that provides a client-based definition of objects that are utilized by the application **440**. The object definition includes a plurality of characteristics associated with each object and utilized by the application to effect interaction with clients over the computer network.

Once the application **440** has been developed using the SDK, the application **440** can be loaded onto one or more network computers and an online session can be established according to the operations shown in the flow diagram box of FIG. **9**. In the first operation, represented by the flow diagram box numbered **910**, a network computer on which the application **440** is loaded connects to a network computer that includes in memory that application interface software. For example, one or more of the client computers **410** of the network system **400** shown in FIG. **4** may have the application **440** loaded in memory and the server computer **420** may include the application interface. In such a case, the client computers **410** establish a communication connection with the server computer **410** over the network **430**.

In the next operation, represented by the flow diagram box numbered **920**, the application **440** registers objects according to the object definitions that are available in the library of the application interface. The application **440** also registers any message filters that will be utilized during the online session, as represented by the flow diagram box numbered **930**.

In the next operation, represented by the flow diagram box numbered **940**, the application **440** defines the session master and assigns ownership of the session master to one of the network computers. The ownership of the session master can be assigned to one computer or can be assigned to plural computers. The application **440** also specifies whether the ownership of the session master is dedicated to a particular computer or whether ownership can migrate to other computers.

During this operation, the application **440** assigns client indices to each of the network computers that will participate in the online session and also establishes the index table described above. The application **440** can be configured such that the first network computer to log onto the server computer will be the session master and also receive an initial index, such as an index of one or zero. It should be appreciated that the initial index may vary. Subsequent network computers to log on will then receive the next available index. After the ownership of the session master or session masters has been established, the online session of the application **440** is commenced, as represented by the flow diagram box numbered **950**.

As noted above, the network computers shown in the block diagram of FIG. **4** comprise nodes of a computer network system **400**. FIG. **10** is a block diagram of a computer in the system **400** of FIG. **4**, illustrating the hardware components included in one of the computers. Those skilled in the art will appreciate that the devices **410** and **420** may all have a similar computer construction, or may have alternative constructions consistent with the capabilities described herein.

FIG. **10** shows an exemplary computer **1000** such as might comprise any of the network computers. Each computer **1000** operates under control of a central processor unit (CPU) **1002**, such as a "Pentium" microprocessor and associated integrated circuit chips, available from Intel Corporation of Santa Clara, Calif., USA. A computer user can input commands and data from a keyboard and computer mouse **1004**, and can view inputs and computer output at a display **1006**. The display is typically a video monitor or flat panel display. The computer **1000** also includes a direct access storage device (DASD) **1008**, such as a hard disk drive. The memory **1010** typically comprises volatile semiconductor random access memory (RAM). Each computer preferably includes a program product reader **1012** that accepts a program product storage device **1014**, from which the program product reader can read data (and to which it can optionally write data). The program product reader can comprise, for example, a disk drive, and the program product storage device can comprise removable storage media such as a magnetic floppy disk, a CD-R disc, a CD-RW disc, or DVD disc.

Each computer **1000** can communicate with the others over a computer network **1020** (such as the Internet or an intranet) through a network interface **1018** that enables communication over a connection **1022** between the network **1020** and the computer. The network interface **1018** typically comprises, for example, a Network Interface Card (NIC) or a modem that permits communications over a variety of networks.

The CPU **1002** operates under control of programming steps that are temporarily stored in the memory **1010** of the computer **1000**. When the programming steps are executed, the computer performs its functions. Thus, the programming steps implement the functionality of the application **440**. The programming steps can be received from the DASD **1008**, through the program product storage device **1014**, or through the network connection **1022**. The program product storage drive **1012** can receive a program product **1014**, read programming steps recorded thereon, and transfer the programming steps into the memory **1010** for execution by the CPU **1002**. As noted above, the program product storage device can comprise any one of multiple removable media having recorded computer-readable instructions, including magnetic floppy disks and CD-ROM storage discs. Other suitable program product storage devices can include magnetic tape and semiconductor memory chips. In this way, the processing steps necessary for operation in accordance with the invention can be embodied on a program product.

Alternatively, the program steps can be received into the operating memory **1010** over the network **1020**. In the network method, the computer receives data including program steps into the memory **1010** through the network interface **1018** after network communication has been established over the network connection **1022** by well-known methods that will be understood by those skilled in the art without further explanation. The program steps are then executed by the CPU **1002** thereby comprising a computer process.

It should be understood that all of the network computers of the network system **400** illustrated in FIG. **4** may have a construction similar to that shown in FIG. **10**, so that details described with respect to the FIG. **10** computer **1000** will be understood to apply to all computers of the system **400**. It should be appreciated that any of the network computers can have an alternative construction, so long as the computer can communicate with the other computers over a network as illustrated in FIG. **4** and can support the functionality described herein.

For example, with reference to FIG. **11**, the client computers **420** can comprise a computer entertainment system, such as a video game system **1100**. FIG. **11** is a block diagram of an exemplary hardware configuration of the video game system **1100**.

The video game system **1100** includes a central processing unit (CPU) **1100** that is associated with a main memory **1105**. The CPU **1100** operates under control of programming steps that are stored in the OS-ROM **1160** or transferred from a game program storage medium to the main memory **1105**. The CPU **1100** is configured to process information and to execute instructions in accordance with the programming steps.

The CPU **1100** is communicatively coupled to an input/output processor (IOP) **1120** via a dedicated bus **1125**. The IOP **1120** couples the CPU **1100** to an OS ROM **1160** comprised of a non-volatile memory that stores program instructions, such as an operating system. The instructions are preferably transferred to the CPU via the IOP **1120** at start-up of the main unit **1100**.

The CPU **1100** is communicatively coupled to a graphics processing unit (GPU) **1110** via a dedicated bus **1115**. The GPU **1110** is a drawing processor that is configured to perform drawing processes and formulate images in accordance with instructions received from the CPU **1100**. For example, the GPU **1110** may render a graphics image based on display lists that are generated by and received from the CPU **1100**. The GPU may include a buffer for storing graphics data. The GPU **1110** outputs images to an audio-visual output device.

The IOP **1120** controls the exchange of data among the CPU **1100** and a plurality of peripheral components in accordance with instructions that are stored in an IOP memory **1130**. The peripheral components may include one or more input controllers **1122**, a memory card **1140**, a USB 1145, and an IEEE 1394 serial bus **1150**. Additionally, a bus **1155** is communicatively coupled to the IOP **1120**. The bus **1155** is linked to several additional components, including the OS ROM **1160**, a sound processor unit (SPU) **1165**, an optical disc control unit **1175**, and a hard disk drive (HDD) **1180**.

The SPU **1165** is configured to generate sounds, such as music, sound effects, and voices, in accordance with commands received from the CPU **1100** and the IOP **1120**. The SPU **1165** may include a sound buffer in which waveform data is stored. The SPU **1165** generates sound signals and transmits the signals to speakers.

The disc control unit **1175** is configured to control a program reader, which can comprise, for example, an optical disk drive that accepts removable storage media such as a magnetic floppy disk, an optical CD-ROM disc, a CD-R disc, a CD-RW disc, a DVD disk, or the like.

The memory card **1140** may comprise a storage medium to which the CPU **1100** may write and store data. Preferably, the memory card **1140** can be inserted and removed from the IOP **1120**. A user can store or save data using the memory card **1140**. In addition, the video game system **1100** is preferably provided with at least one hard disk drive (HDD) **1180** to which data may be written and stored.

A data I/O interface, such as an IEEE 1394 serial bus **1150** or a universal serial bus (USB) **1145** interface, is preferably communicatively coupled to the IOP **1120** in order to allow data to be transferred into and out of the video game system **1100**, such as to the network **430** of FIG. **4**.

The system and method described above improve on the situation where a network user of an application, such as a game player, is performing as an Integrated Server (IS) for the application, thereby maintaining an application environment, so that the application would end for a conventional implementation of the application when that IS user wishes to log off. As described above, some applications (such as multi-user gaming applications) alternatively permit the functions (and data) of the departing user to be migrated from the departing user to a different user, who will continue with the online session and will take over the duties of the IS. This type of hand-off is typically rather cumbersome and might not be accomplished smoothly. In the case of a gaming environment, for example, a departing player might abruptly disappear from the game environment, thereby disrupting the gaming experience of the other players. The multi-user application in accordance with the invention permits continued use of the application, even with users departing and joining, by notifying all user machines when another user has departed from the session. Suitable adjustments can be made for a more pleasing application environment. The notification occurs through a Disconnect function that ensures proper IS operation and communications. That is, an application server or IS of the application can broadcast a message to all clients in an application environment to notify them that a user has departed or has joined, and can ensure appropriate functionality, if necessary, of the users.

For a system with a network device operating as an Integrated Server (IS) as described above by serving application data to other users, the failure or departure of an IS from the application environment is responded to by assigning a different user as a new IS. The application can assign a new IS by automatically carrying out a replacement process, or by sending a broadcast message to all users and awaiting replies. For automatic selection, the application can assign the new IS in accordance with considerations that include the bandwidth available to the user, the geographic location of the potential new IS, a user's indicated preference for consideration as an IS, the technical specification and resources available at the user's machine, or through a random selection process. If the application is designed so that it sends a broadcast message, then the message will typically solicit voluntary agreement to operate as the new IS.

In addition, a new user who logs in to the system after an IS failure and who otherwise might have been registered with the failed IS can instead be diverted to a different IS, reducing the workload of the group being served by the now-unavailable IS. In this way, newcomers who wish to join the application environment of an IS can instead be moved to a different IS and different user group. Alternatively, the application can respond to a failed IS by dissolving or disbanding the online session of the group administered by the unavailable IS and forming a new online group with a new IS. These alternatives

can be selected by an application developer who is configuring an application for operation in accordance with the present invention.

If an individual user leaves during an online session, the result can be somewhat more problematic. In the online gaming context, for example, a certain minimum number of users (players) are required for a game to proceed. In accordance with the invention, the application can respond by sending a message to other users on the network, inviting others to join the online session and participate in the multi-user application (such as a game). Alternatively, the application can be configured so as to invoke an Artificial Intelligence module to carry out the duties of the Integrated Server.

The present invention has been described above in terms of a presently preferred embodiment so that an understanding of the present invention can be conveyed. There are, however, many configurations for the system and application not specifically described herein but with which the present invention is applicable. The present invention should therefore not be seen as limited to the particular embodiment described herein, but rather, it should be understood that the present invention has wide applicability with respect to multi-user applications generally. All modifications, variations, or equivalent arrangements and implementations that are within the scope of the attached claims should therefore be considered within the scope of the invention.

What is claimed is:

1. A method of managing participants in an online session, comprising:

participating in an online session including a plurality of participants communicatively linked to one another via a gaming network, wherein each of the participants includes a computing device;

detecting that a first participant responsible for certain managerial functionality associated with the online session has disconnected from the online session, wherein detecting that the first participant has disconnected from the online session occurs when an update message periodically broadcast by the first participant is not received within a predetermined amount of time;

broadcasting a notification to the remaining participants from the plurality of participants in the online session that the first participant has disconnected from the online session, wherein the notification is broadcast following detection of the first participant having disconnected from the online session; and

accepting a new participant to the online session to replace the disconnected first participant.

2. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the functionality associated with the first participant and re-assigned to another participant in the online session is re-assigned to the new participant.

3. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a condition of the gaming network.

4. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises the geographical location of one or more of the participants in the online session.

5. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a hardware specification of one or more of the participants in the online session.

6. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a preference specified by a user of one of the participant computing devices.

7. The method of claim **1**, further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the re-assignment of functionality associated with the first participant is random.

8. A system for managing participants in an online session, the system comprising:

- a gaming network for establishing the online session amongst a plurality of participant computing devices, each of the plurality of participant computing devices in the online session are communicatively linked to one another via the gaming network;
- a first participant computing device from the plurality of participant computing devices, wherein the first participant computing device is responsible for certain managerial functionality associated with the online session, the first participant computing device configured to periodically broadcast an update message to the other participant computing devices in the online session; and
- a second participant computing device configured to:
  - receive the periodically broadcast update message, wherein the second participant computing device will determine that the first participant computing device has disconnected from the online session if the update message is not received within a predetermined period of time,
  - broadcast a notification to the remaining participant computing devices from the plurality of participant computing devices in the online session that the first participant computing device has disconnected from the online session, wherein the notification is broadcast following the determination that first participant computing device has disconnected from the online session, and
  - accept a new participant computing device to replace the disconnected first participant computing device.

9. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the functionality associated with the first participant computing device and re-assigned to another participant computing device in the online session is re-assigned to the new participant computing device.

10. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the one or more determinations comprises a condition of the gaming network.

11. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the one or more determinations comprises the geographical location of one or more of the participant computing devices in the online session.

12. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the one or more determinations comprises a hardware specification of one or more of the participant computing devices in the online session.

13. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the one or more determinations comprises a preference specified by a user of one of the participant computing devices.

14. The system of claim **8**, wherein the second participant computing device is further configured to re-assign the functionality associated with the first participant computing device to another participant computing device in the online session based on one or more determinations made by one of the remaining participant computing devices in the online session and wherein the re-assignment of functionality associated with the first participant computing device is random.

15. A computer readable storage medium having embodied thereon a program, the program being executable by a computer to perform a method for managing participants in an online session, the method comprising:

- detecting that a first participant from a plurality of participants in the online session has disconnected from the online session, wherein each of the participants in the online session includes a computing device communicatively linked to one another via a gaming network and the first participant is responsible for certain managerial functionality associated with the gaming network, wherein detecting that the first participant has disconnected from the online session occurs when an update message periodically broadcast by the first participant is not received within a predetermined amount of time;
- broadcasting a notification to the remaining participants from the plurality of participants in the online session that the first participant has disconnected from the online session, wherein the notification is broadcast following detection of the first participant having disconnected from the online session; and
- accepting a new participant to the online session to replace the disconnected first participant.

16. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the functionality associated with the first participant and re-assigned to another participant in the online session is re-assigned to the new participant.

17. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a condition of the gaming network.

18. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises the geographical location of one or more of the participants in the online session.

19. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a hardware specification of one or more of the participants in the online session.

20. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the one or more determinations comprises a preference specified by a user of one of the participant computing devices.

21. The computer-readable storage medium of claim **15**, the method further comprising re-assigning the functionality associated with the first participant to another participant in the online session based on one or more determinations made by one of the remaining participants in the online session and wherein the re-assignment of functionality associated with the first participant is random.

* * * * *

EXHIBIT 90

US007792902B2

(12) **United States Patent**
Chatani et al.

(10) **Patent No.: US 7,792,902 B2**
(45) **Date of Patent: Sep. 7, 2010**

(54) MANAGING PARTICIPANTS IN AN ONLINE SESSION

(75) Inventors: **Masayuki Chatani**, Foster City, CA (US); **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America LLC**, Foster City, 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.: **11/375,526**

(22) Filed: **Mar. 13, 2006**

(65) **Prior Publication Data**

US 2006/0173958 A1 Aug. 3, 2006

Related U.S. Application Data

(63) Continuation of application No. 10/211,128, filed on Jul. 31, 2002.

(60) Provisional application No. 60/381,736, filed on May 17, 2002.

(51) **Int. Cl.**
G06F 15/16 (2006.01)
A63F 9/14 (2006.01)

(52) **U.S. Cl.** **709/205**; 709/209; 463/42

(58) **Field of Classification Search** 709/208, 709/204, 205, 209; 463/62
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,630,757 A		5/1997	Gagin et al.	
5,682,139 A	*	10/1997	Pradeep et al.	340/539.13
5,704,032 A		12/1997	Badovinatz et al.	
5,838,909 A	*	11/1998	Roy et al.	709/209

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1125617	8/2001

(Continued)

OTHER PUBLICATIONS

Festa et al., "Netscape Alumni to Launch P2P Company," Aug. 2, 2001.

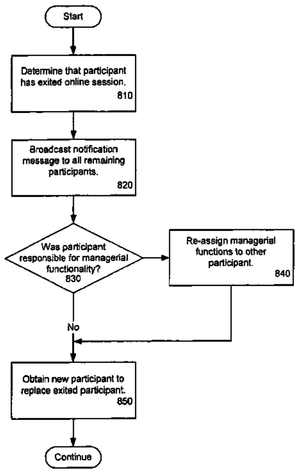
(Continued)

Primary Examiner—Douglas B Blair
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. The application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

27 Claims, 11 Drawing Sheets


Start
Determine that participant has exited online session. 810
Broadcast notification message to all remaining participants. 820
Was participant responsible for managerial functionality? 830
Re-assign managerial functions to other participant. 840
No
Obtain new participant to replace exited participant. 850
Continue

U.S. PATENT DOCUMENTS

5,841,980	A	11/1998	Waters et al.	
5,893,106	A	4/1999	Brobst et al.	
5,956,485	A	9/1999	Perlman	
5,984,787	A	11/1999	Redpath	
5,987,376	A	11/1999	Olson et al.	
6,041,312	A	3/2000	Bickerton et al.	
6,152,824	A	11/2000	Rothschild et al.	
6,154,782	A	11/2000	Kawaguchi et al.	
6,219,045	B1	4/2001	Leahy et al.	
6,311,209	B1 *	10/2001	Olson et al.	709/204
6,487,678	B1	11/2002	Briskey et al.	
6,560,636	B2 *	5/2003	Cohen et al.	709/203
6,561,811	B2 *	5/2003	Rapoza et al.	434/236
6,607,444	B2	8/2003	Takahashi et al.	
6,631,412	B1 *	10/2003	Glasser et al.	709/224
6,676,521	B1	1/2004	La Mura et al.	
6,748,420	B1	6/2004	Quatrano et al.	
6,761,636	B2 *	7/2004	Chung et al.	463/42
6,931,446	B1	8/2005	Cox et al.	
7,003,550	B1 *	2/2006	Cleasby et al.	709/205
7,016,942	B1	3/2006	Odom	
7,018,295	B2 *	3/2006	Sakaguchi et al.	463/42
7,056,217	B1	6/2006	Pelkey et al.	
7,107,312	B2 *	9/2006	Hackbarth et al.	709/204
7,290,264	B1 *	10/2007	Powers et al.	719/315
7,587,465	B1 *	9/2009	Muchow	709/209
2001/0009868	A1 *	7/2001	Sakaguchi et al.	463/42
2001/0044339	A1 *	11/2001	Cordero et al.	463/42
2002/0035604	A1	3/2002	Cohen et al.	
2002/0049086	A1 *	4/2002	Otsu	463/42
2002/0062348	A1	5/2002	Mashiro	
2002/0075844	A1	6/2002	Hagen	
2002/0133707	A1 *	9/2002	Newcombe	713/183
2003/0073494	A1 *	4/2003	Kalpakian et al.	463/42
2003/0217135	A1	11/2003	Chatanl	
2003/0217158	A1	11/2003	van Datta	
2004/0117443	A1 *	6/2004	Barsness	709/204
2005/0251577	A1	11/2005	Guo	
2005/0259637	A1	11/2005	Chu	
2006/0100020	A1 *	5/2006	Kasai	463/42
2006/0253595	A1	11/2006	van Datta	
2006/0288103	A1	12/2006	Gobara	
2007/0058792	A1	3/2007	Chaudhari	
2007/0076729	A1	4/2007	Takeda	
2007/0165629	A1	7/2007	Chaturvedi	
2007/0191109	A1	8/2007	Crowder	
2007/0208748	A1	9/2007	Li	
2008/0280686	A1 *	11/2008	Dhupelia et al.	463/42
2009/0077245	A1	3/2009	Smelyansky	
2009/0138610	A1	5/2009	Gobara	
2009/0240821	A1	9/2009	Juncker	

FOREIGN PATENT DOCUMENTS

WO	WO 00/05854	2/2000
WO	WO 01/82678	11/2001
WO	WO 02/35769	5/2002

OTHER PUBLICATIONS

Diot et al, "A Distributed Architecture for Multlplayer Interactive Applications on the Internet," IEEE vol. 13, Issue 4, Aug. 1999.
European Search Report for EP 03 72 1413, Jun. 30, 2005.
Cisco Systems, Inc., "Network Flow Management: Resource Reservation for Multimedia Flows," Mar. 19, 1999.
"Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Mar. 23, 2007).
"Brief for Appellee," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (May 21, 2007).
"Reply Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Jun. 4, 2007).
"Petition for Panel Rehearing," In Re Masayuki Chatani and Glen Van Datta, Appeal From the United States Patent and Trademark Office, Boar of Patent Appeals and Interferences, In the United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 10/211,128), Jan. 3, 2008.
"In Re Masayuki Chatani and Glen Van Datta," United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 11/211,128), Nov. 19, 2007.
Hagsand O: "Interactive Multiuser VEs in the DIVE System" IEEE Multimedia, IEEE Service Center, New York, NY, US vol. 3, No. 1, Mar. 21, 1996, pp. 30-39, XP000582951 ISSN:1070-986X.

* cited by examiner

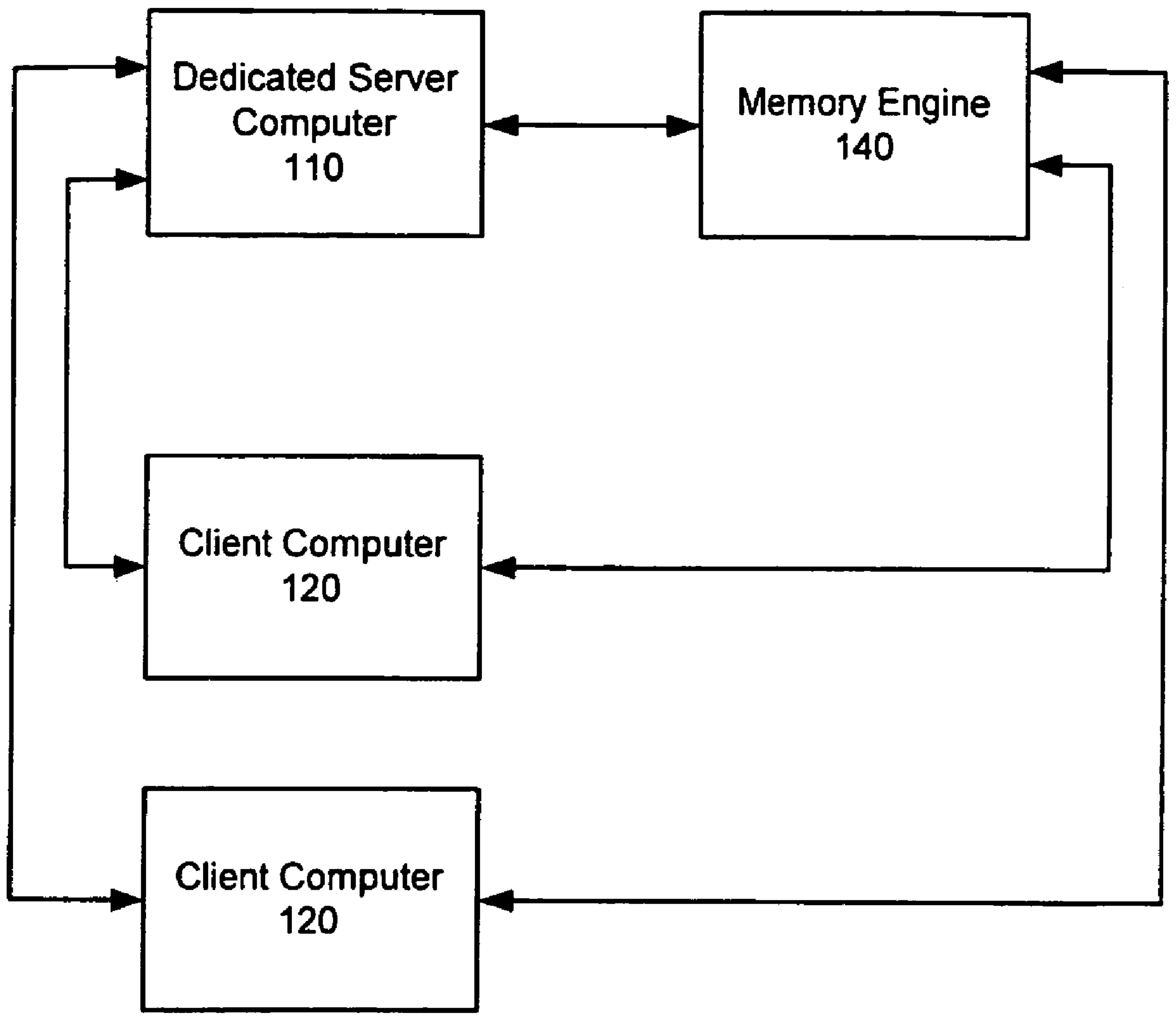


Figure 1

PRIOR ART

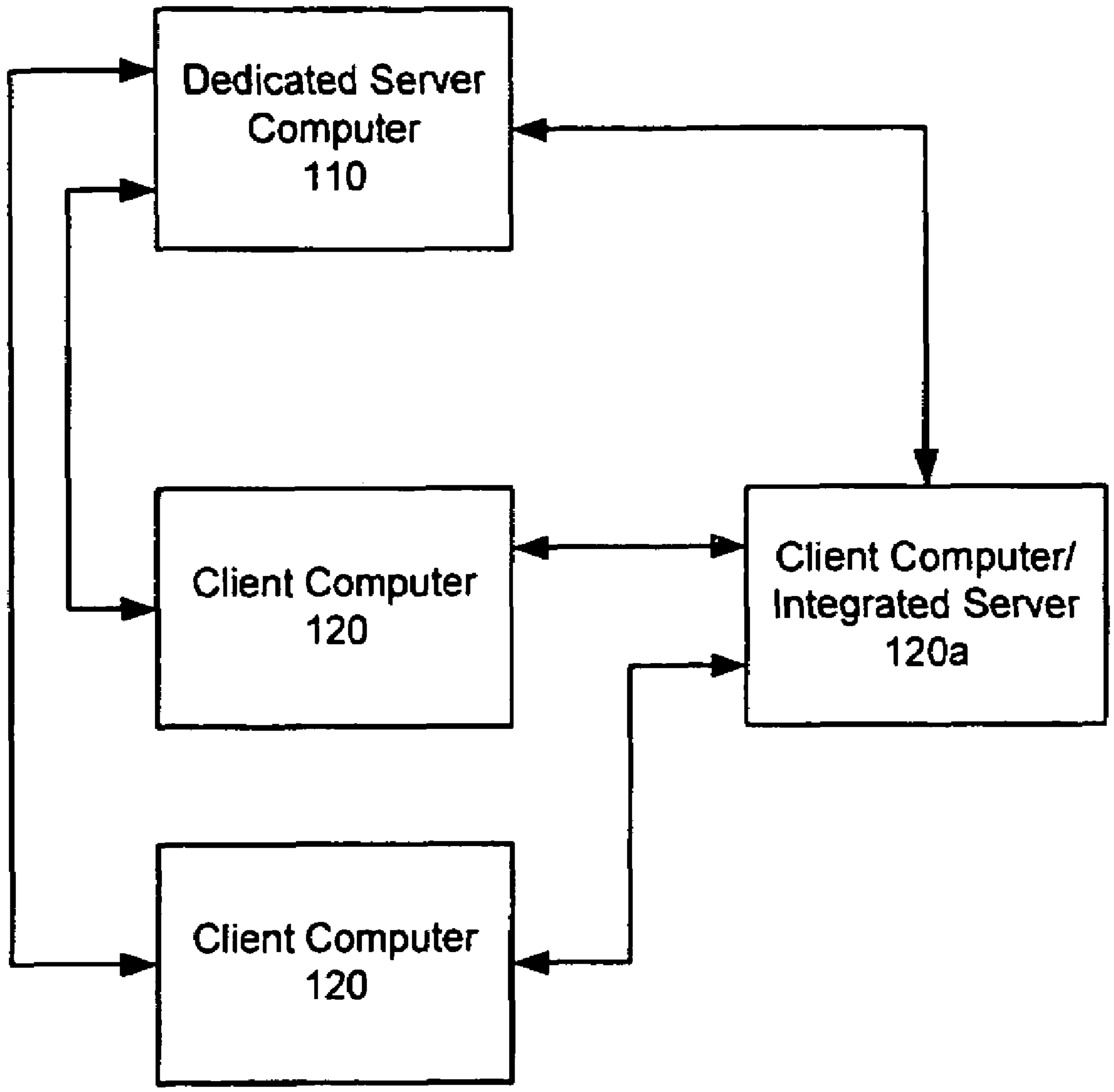

Dedicated Server Computer 110
Client Computer 120
Client Computer/ Integrated Server 120a
Client Computer 120

Figure 2

PRIOR ART

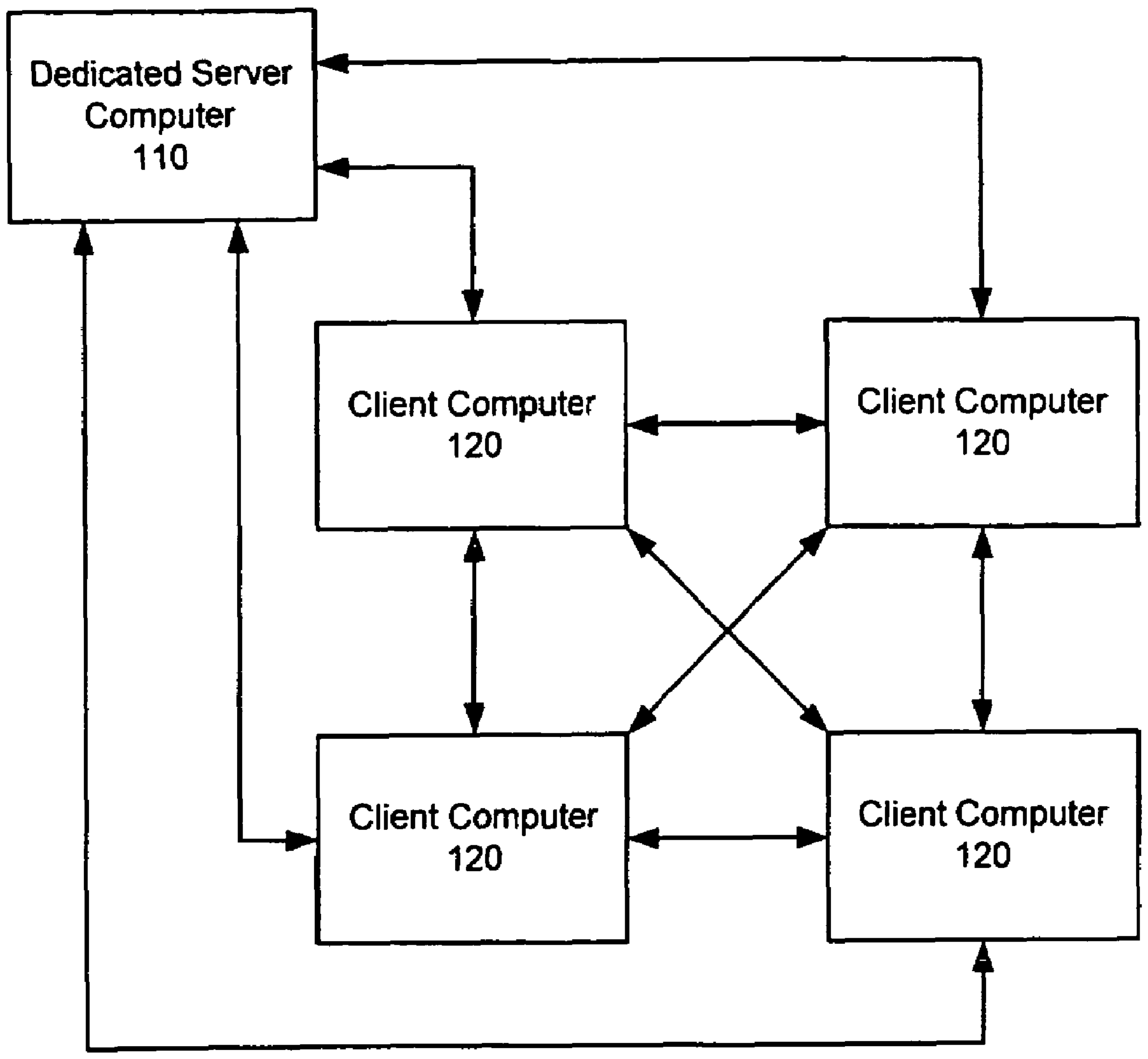


Figure 3

PRIOR ART

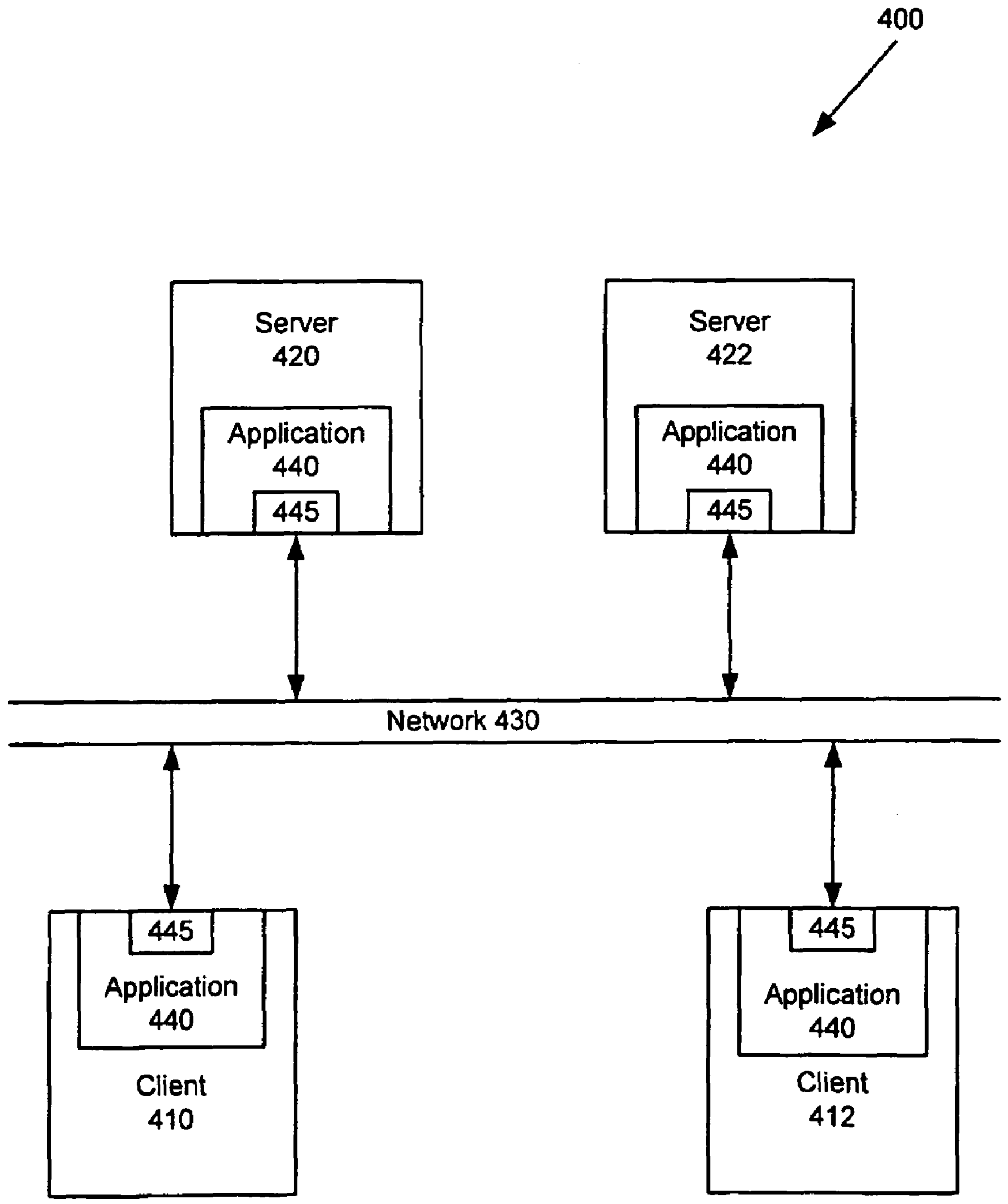

400
Server 420
Application 440
445
Server 422
Application 440
445
Network 430
445
Application 440
Client 410
445
Application 440
Client 412

Figure 4

500

Index	Session Master	Active Port/Protocol Type
1	Owns C1	80/TCP, 90/UDP
2	Owns C2, C3	85/TCP, 95/UDP
3	-	
-	-	
-	-	
-	-	

Figure 5

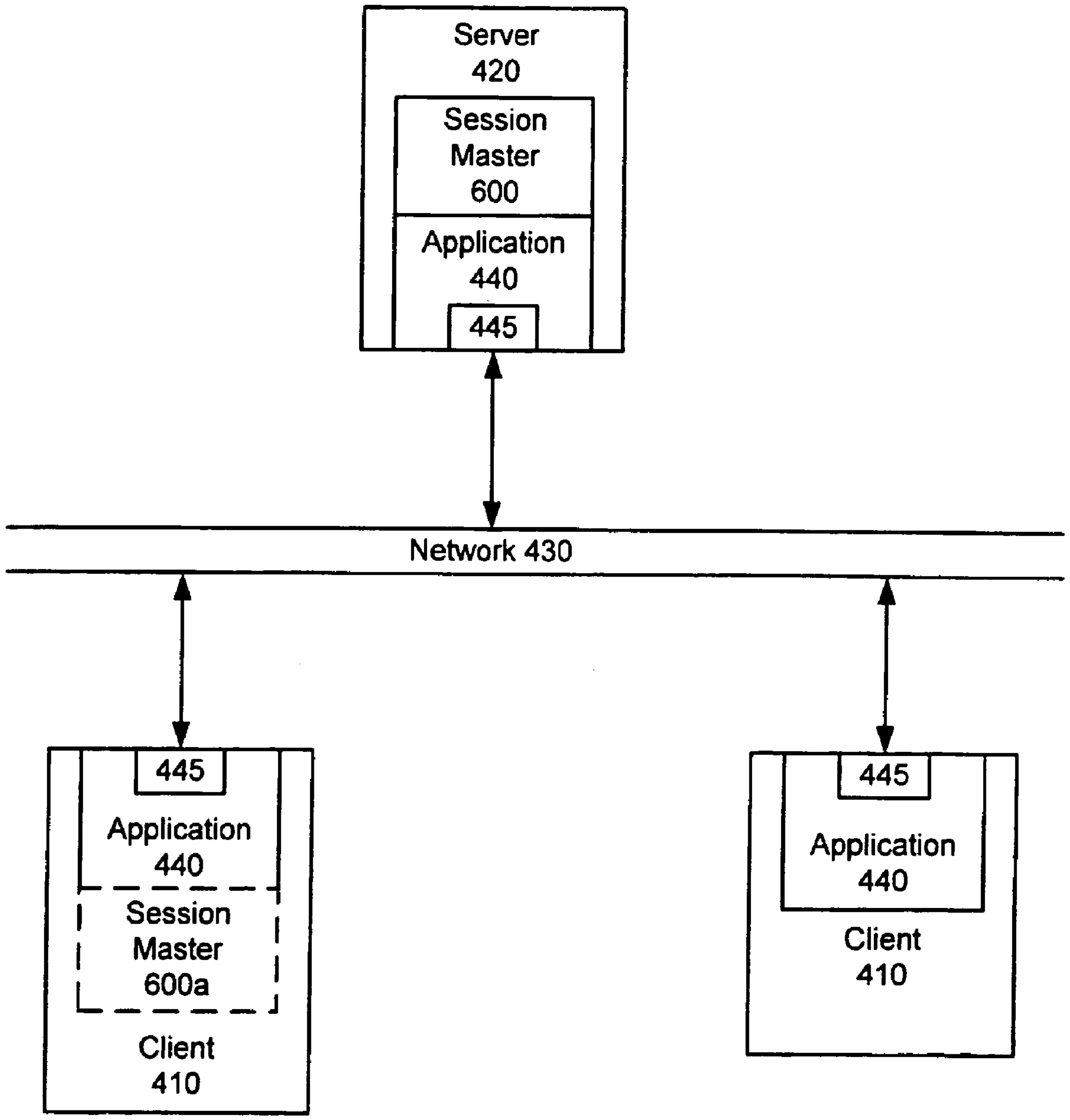

Server
420
Session
Master
600
Application
440
445
Network 430
445
Application
440
Session
Master
600a
Client
410
445
Application
440
Client
410

Figure 6

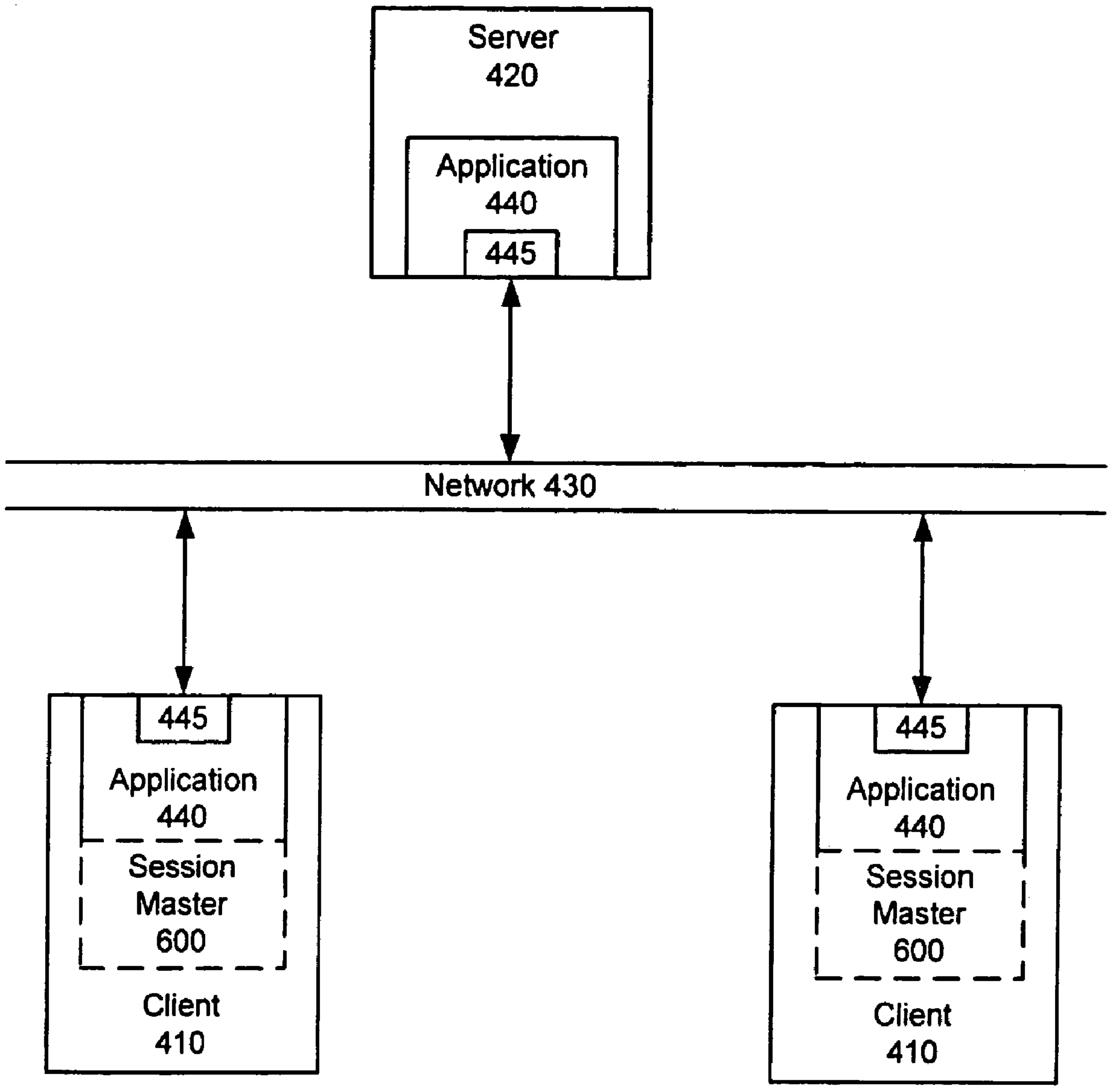

Server
420
Application
440
445
Network 430
445
Application
440
Session
Master
600
Client
410
445
Application
440
Session
Master
600
Client
410

Figure 7

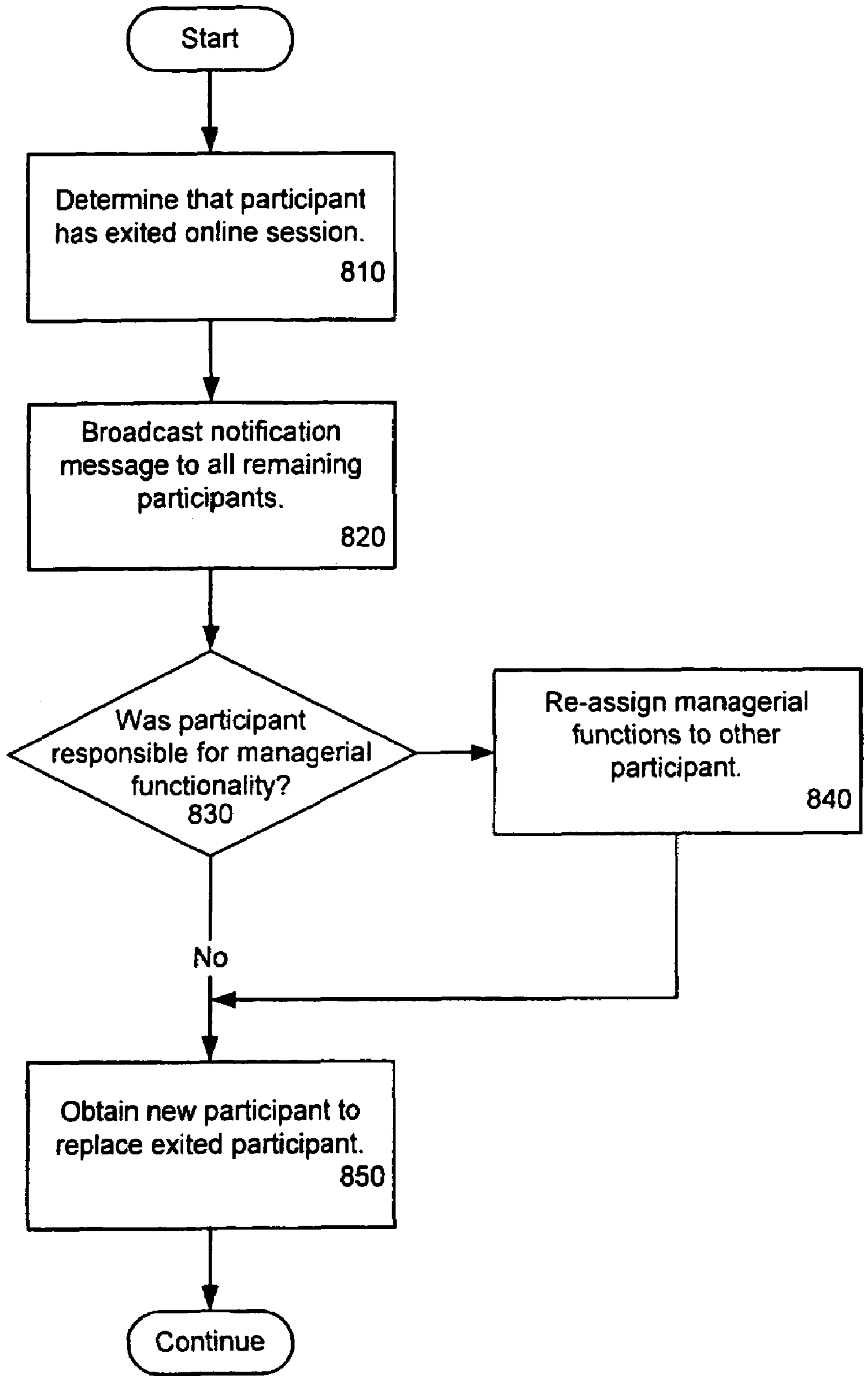

Start
Determine that participant has exited online session.
810
Broadcast notification message to all remaining participants.
820
Was participant responsible for managerial functionality?
830
Re-assign managerial functions to other participant.
840
No
Obtain new participant to replace exited participant.
850
Continue

Figure 8

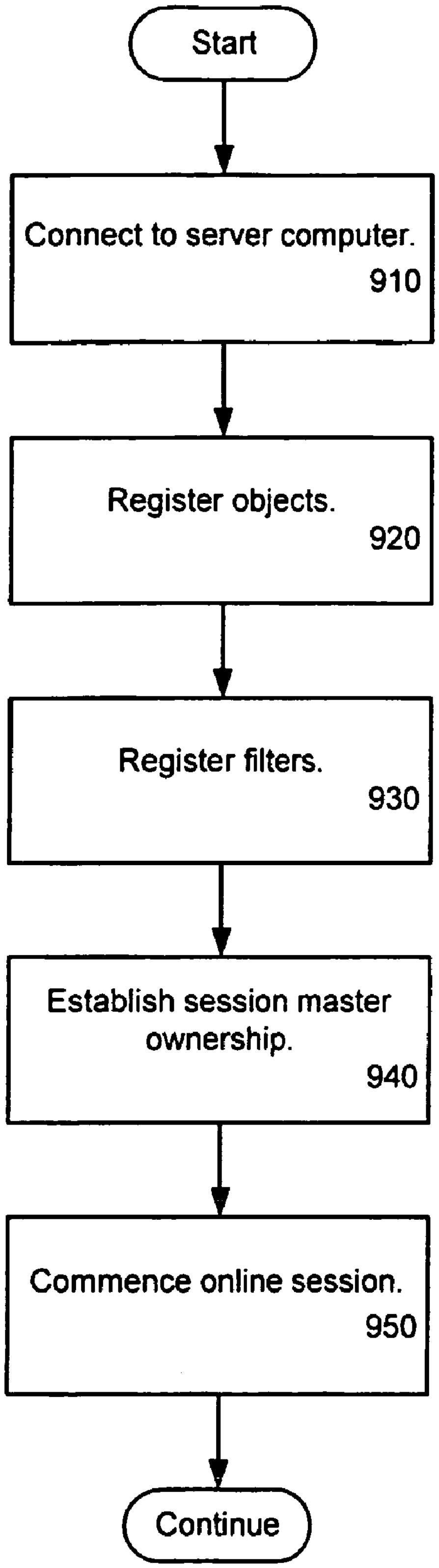

Start
Connect to server computer.
910
Register objects.
920
Register filters.
930
Establish session master ownership.
940
Commence online session.
950
Continue

Figure 9

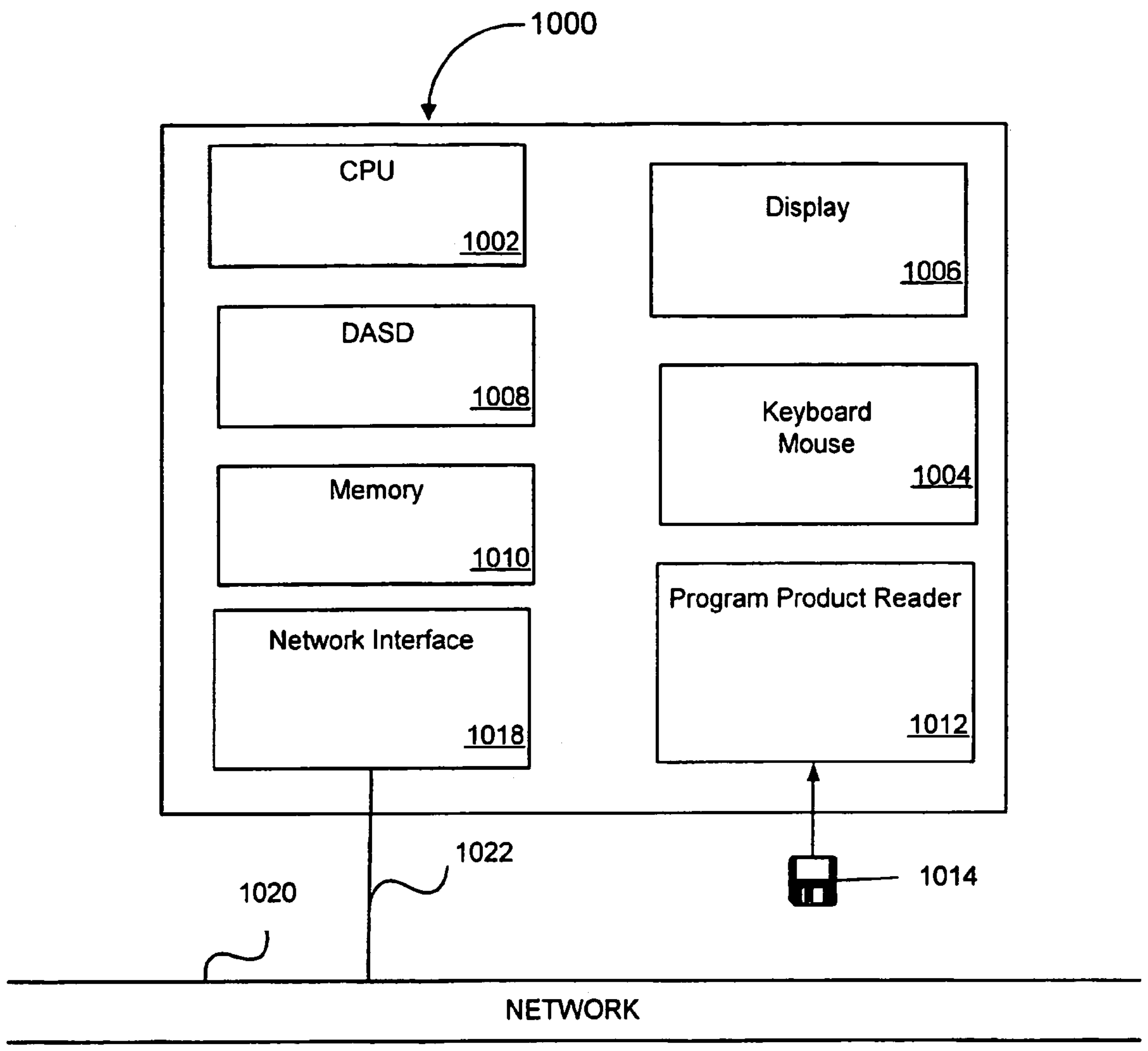

1000
CPU
1002
DASD
1008
Memory
1010
Network Interface
1018
Display
1006
Keyboard
Mouse
1004
Program Product Reader
1012
1022
1020
1014
NETWORK

Figure 10

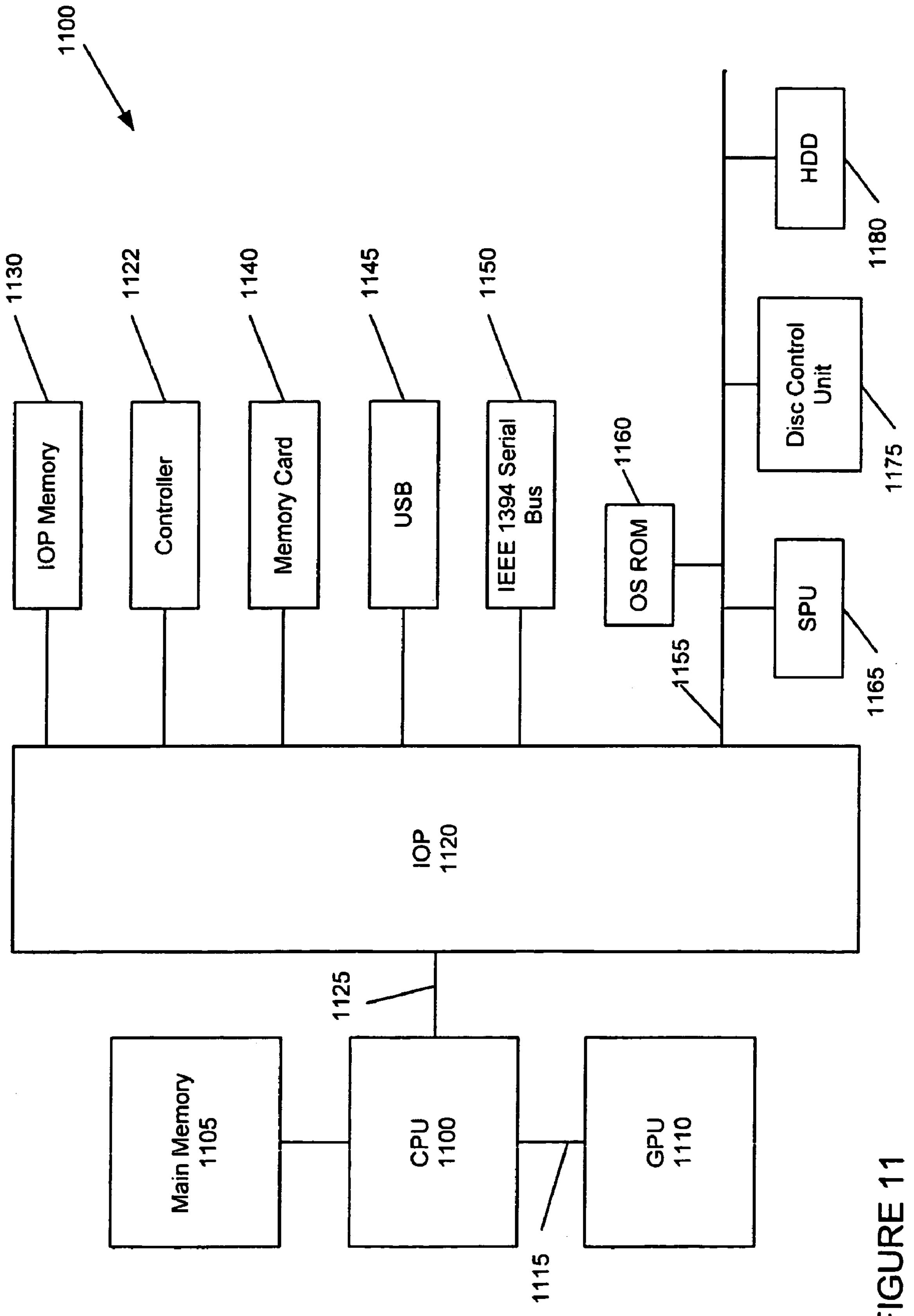

1100
IOP Memory
1130
Controller
1122
Memory Card
1140
USB
1145
IEEE 1394 Serial Bus
1150
OS ROM
1160
1155
SPU
1165
Disc Control Unit
1175
HDD
1180
IOP
1120
1125
Main Memory
1105
CPU
1100
GPU
1110
1115

FIGURE 11

MANAGING PARTICIPANTS IN AN ONLINE SESSION

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation and claims the priority benefit of U.S. patent application Ser. No. 10/211,128 filed Jul. 31, 2002 and entitled “Dynamic Player Management,” which claims the priority benefit of U.S. provisional patent application No. 60/381,736 filed May 17, 2002 and entitled “Dynamic Player Management.” The disclosure of these commonly owned applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to computer networks and, more particularly, to an application that is run by multiple users on a computer network.

2. Description of the Related Art

Computer networks, such as local area networks and the Internet, are increasingly being used as the backbone for various transactions and interactions between parties. From online banking, where bank customers can initiate financial transactions on a computer network, to online gaming, where garners can participate in various games over the Internet, service providers are increasingly providing a variety of services over computer networks. There are currently a variety of different computer network configurations that facilitate the transactions and interactions that take place.

One type of configuration is a classic client-server configuration, such as is illustrated in FIG. **1**. In this configuration, a dedicated server computer **110** is communicatively linked to one or more client computers **120** over a network, such as through the Internet. The client computer **120** makes service requests to the server computer **110** and the server computer **110** fulfills the request by transmitting data to the requesting client computer **120** over the network. The server computer **110** can be connected to a data storage device or to other computer devices that facilitate transactions between the client and server computers. One characteristic of the client-server configuration is that the client computers cannot communicate directly with one another, as the client computers are limited to communicating with the server computer.

For example, where the client-server configuration is operated in an online gaming environment, the server computer **110** can be responsible for maintaining the various states that are associated with the online game. The server computer can be connected to other computers, such as a memory engine **140** that maintains one or more instances of a game, while the server computer **110** manages administrative matters such as player matching and account management. A game player on the client computer **120** can log onto the server computer **110** and receive a list of available games and participating players. The player chooses a game to start or join, thereby identifying a memory engine with which the player's computer establishes a client-server connection. In this manner, the server computer **110** and the memory engine **140** collectively administer the gaming environment for one or more client computers **120**.

Another type of configuration is referred to as an integrated server configuration, such as is shown in FIG. **2**. This configuration includes a dedicated server computer **110** and one or more client computers **120** that are each connected to the server computer **110** over a computer network. As in the

previously-described configuration, the server computer **110** serves data to the client computers **120**. However, one of the client computers **120**, such as the client computer **120***a*, functions as an integrated server in that the client computer **120***a* can serve data to the other client computers **120**. In an online gaming environment, the server computer **110** can perform administrative functions, such as player matching, account management, and chat room management, while the client computer/integrated server **120***a* can perform the function of the previously-described memory engine.

In yet another type of communication configuration, the various computers are arranged in a peer-to-peer configuration, such as is shown in FIG. **3**. In a peer-to-peer configuration, each of the computers can communicate with the others, so that all of the computers function as “peers.” In one form of the peer-to-peer configuration, a dedicated server **110** is communicatively connected to a plurality of client computers **120** over a network. An online session is initially established by each of the client computers **120** connecting to an administrative computer, such as the server computer **110**. The client computers **120** are then communicatively connected to one another so that each of the client computers **120** has the ability to both serve and receive data to and from any of the other client computers **120**. In addition, each client computer **120** can operate in a client-server relationship with the dedicated server **110**. Those skilled in the art will appreciate that there are other communication configurations in addition to the configurations described above.

The various configurations described above enable computer users to interact over a computer network, such as in an online game environment where game players can play computer games on a computer network. In such a scenario, at least one of the computers typically functions as a game manager that manages various aspects of the game, such as coordinating the number of players, keeping track of game state, and sending out updates to the users regarding game state. It can be appreciated that continuity of game play can be highly dependent on all of the users in a game continuing to play throughout the entire game period. Game play can be interrupted or even halted if one of the game players exits during the middle of a game, particularly where the exited player was managing a portion of the game.

For example, sports games typically have a fixed start and a fixed finish for the game, with at least two players competing in a game. In current configurations, there are often several players that participate in an online sporting contest, with each one of the players assuming the role of a player on a sporting team. For example, in an online football game, players can assume the roles of quarterback, receiver, defensive back, running back, etc. If one of the players were to suddenly leave during the middle of the game, then the game play would be interrupted or halted. This could also be the case in other types of games, where continuity of game play is dependent on each of the players in the game environment continuing to play throughout a particular scenario.

Unfortunately, current multi-user applications are not configured to account for when a participant in an online session suddenly or unexpectedly leaves the online session. If a player does leave an online session, then the session is unduly interrupted or terminated. In view of the foregoing, there is a need for a multi-user application that overcomes the aforementioned shortcomings.

SUMMARY OF THE INVENTION

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. In accordance with one aspect of the invention, the application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

Other features and advantages of the present invention should be apparent from the following description of the preferred embodiment, which illustrates, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of a computer network arranged in a client-server network communication configuration.

FIG. **2** is an illustration of a computer network arranged in an integrated network communication server configuration.

FIG. **3** is an illustration of a computer network arranged in a peer-to-peer network communication configuration.

FIG. **4** is an illustration of a computer network system on which is run a multi-user application configured in accordance with the present invention.

FIG. **5** is an illustration of a data structure that includes computer index and session master information for the multi-user application.

FIG. **6** is an illustration of a computer network system wherein the multi-user application is arranged in a first type of communication configuration.

FIG. **7** is an illustration of a computer network system wherein the multi-user application is arranged in another type of communication configuration.

FIG. **8** is a flow diagram that represents a process of managing the exiting of a participant in an online session of a multi-user application.

FIG. **9** is a flow diagram that illustrates the operating steps associated with the multi-user application establishing an online session.

FIG. **10** is a block diagram of a computer in the network illustrated in FIG. **4**, illustrating the hardware components.

FIG. **11** is a block diagram of a computer entertainment system in the network illustrated in FIG. **4**, illustrating the hardware components.

DETAILED DESCRIPTION

FIG. **4** is a block diagram of a computer network system **400** comprised of one or more network devices including one or more client computers **410**, **412** and one or more dedicated server computers **420**, **422**, which are nodes of a computer network **430**. Thus, some of the network computers are configured as servers and some are configured as clients. The computer network **430** may comprise a collection of interconnected networks, such as the Internet, and may include one or more local area networks at each of the nodes **410**, **412**, **420**, **422**. As used herein, the term "Internet" refers to a collection of interconnected (public and/or private) networks that are linked together by a set of standard communication protocols to form a global, distributed network.

The client computers **410**, **412** can transmit requests for data over the network **430** to one of the server computers **420**, **422**, which are configured to serve data over the network **430** to the client computers in a well-known manner. The server computers **420**, **422** can include or can be communicatively linked to each other and to other servers, such as a data base server and/or an application server, as will be known to those skilled in the art. Although FIG. **4** shows only two client computers **410**, **412** and two server computers **420**, **422**, it should be appreciated that the network system **400** could include any number of client computers **410**, **412** and server computers **420**, **422**. The server computers **420**, **422** and client computers **410**, **412** are sometimes referred to collectively herein as network computers.

The network system **400** supports a multi-user application **440** comprised of a computer program with which multiple users can interact in online sessions using network devices (such as the client computers **410**, **412**) that are linked to the computer network **430**. The application **440** is installed at each of the client computers, meaning that an operational instance of the application is stored in memory of each of the client computers **410**, **412** that run (execute) the application **440**. Each server computer that will be participating in an online session of the multi-user application also stores an instance of the application **440**. For purposes of this description, the first server computer **420** will be assumed to be a server for the multi-user application being executed by the client machines **410**, **412**, although both servers **420**, **422** are shown with installed applications **440**. An exchange of data occurs between instances of the application **440** during execution and is enabled through the establishment of network sockets **445** at each of the network computers. The sockets are represented in FIG. **4** as boxes at each respective network computer. Those skilled in the art will understand that a network socket is one end of a multi-way communication link between two or more programs that run on the network system **400**.

The application **440** can be run on the network devices of the network system **400** according to a variety of communication configurations and the responsibilities for various application-related processes can be assigned to different computing devices of the network **430**, as described in more detail below. An application development interface is preferably used to develop the application **440**, as is also described in more detail below. The application can be operated such that the associated network computer can use a communications configuration to implement any of the communication modes illustrated in FIG. **1**, FIG. **2**, and FIG. **3**.

The multi-user application **440** can be any type of application that a user can run on a network computer that is linked to the computer network **430**. When the application **440** is run on a client computer **410**, **412**, the user can interact with other users through other network computers that are also running the application **440**. The server computer **420** can function as a central network "meeting point" through which the users can establish contact, maintain data, and initiate an online session of the application **440**. Typically, the application **440** causes the network device in which it is operating to establish communications with another network device, such as the devices **410**, **412**, **420**, thereby initiating an online session.

During the online session, the network computers will interact and exchange data pursuant to the programmed features of the application **440**.

When the application **440** is launched and an online session is established among suitably configured computers, the application enables the computers to interact in a variety of configurations. Throughout this description, the application **440** is sometimes described in an online-gaming scenario, wherein the application **440** comprises a computer game that multiple users can access and run using the client computers **410**, **412**. In such a case, the application **440** establishes an online session comprised of a game in which the network computers participate. However, it should be appreciated that the application **440** may also relate to other scenarios besides gaming, such as, for example, online banking or online travel planning, that involve interactions between multiple computers on a computer network.

When an application **440** executes, it identifies a session master, which is a network computer that performs a variety of managerial and administrative functions for the application with respect to interactions between computers that occur during an online session. An online session of the application uses a registration or logon process with a data store containing information such as user identification. The logon process authorizes further participation in the network environment of the application. Preferably, the session master function is assigned when a client computer running the application **440**, such as the client computer **410**, logs onto the server computer **420** to initiate an online session. The application itself, however, determines the details of when and how such assignments are made, so that a variety of session master assignment schemes can be implemented without departing from the teachings of the present invention.

The operating instance of the application on the client computer **410** that initiates an online session of the application is referred to as the host computer. The application at the host computer assigns the session master function to either the server computer **420** or to the host computer **410**. As new client computers log-on (register) with the server computer **420** to join the online session, the server computer **420** notifies the new clients of the already-assigned identity of the session master computer.

As described more fully below, the session master functionality enables a smooth transition between the various network communication configurations in which the application **440** can operate. The session master function also enables the application **440** to concentrate responsibility for application-related tasks in a specific network computer, or to distribute such responsibility among two or more network computers. The assignment of tasks can be performed by an instance of the application **440** on one of the network computers, at the same time when the session master function is assigned, and the session master tasks can be assigned to one or more of the computers on the network **430** for providing the requisite functionality. The computer or computers that are assigned responsibility of the session master are referred to herein as the "owners" of the respective session master functions. References to a solitary session master will be understood to apply to a group of computers, if those computers are collectively performing the session master functions. Thus, the assignment of session master tasks is performed in the manner specified by the application, in accordance with the dictates of the application developer.

One category of responsibilities assigned to the session master relates to application-specific functions, which are functions that are peculiar to the particular type of application **440** being executed. For example, if the application **440** is a

game-type application, then the session master or a group of session masters can keep track of game-type data, such as the game score and the time remaining in the game, and can perform game functions, such as terminating the online session when a game ends. A session master computer can also be assigned the responsibility of keeping track of specific game data, such as the state of an object in the game environment, such as a football, aircraft, ocean, tree, and so forth. Each of these responsibilities can be concentrated in a single session master computer or can be divided up among several session master computers, in accordance with the operation of the application.

The host computer performs managerial functions related to the computers that are participating in the online session. For example, whenever a network computer joins an online session of the application **440**, the host computer assigns an identification index number to the computer joining the session. The host computer maintains a list of the identification index numbers and their associated network computers. The index number is used when sending messages and is also used to maintain ownership records regarding the session master functionality.

As noted above, there can be more than one session master in an online session. The way in which a session master is assigned can be determined by the application in accordance with the operation of the application. The application **440** can also assign the session master with responsibility for sending out update messages to update the network computers regarding the status of all network computers that are participating in the online session. This responsibility entails the session master notifying the participating network computers when a new network computer joins the online session, or when a current participant exits the online session of the application **440**, as described more fully below.

The host computer that assigns the aforementioned index number to each of the computers also maintains a list of all network computers that are participating in the online session. The application **440** then keeps track of session master ownership according to the index number assigned to the computer. To keep track of the index number and responsibility assignments, the application **440** can maintain a data structure such as in the form of a table comprised of a network computer index list, such as the table **500** shown in FIG. **5**. The table **500** contains an index number associated with each network computer that is participating in the online session, and also contains an indication of whether the network computer owns the session master function. The index list data structure comprising the table **500** preferably also specifies the communication protocol that is being used for each network computer. FIG. **5** shows that different session master tasks (C**1**, C**2**, C**3**) can be owned by different network computers.

In addition to specifying the communication protocol, the data structure also specifies, for each network computer, the port for which the communication protocol is associated. Each instance of the application **440** enables the associated network computer that is participating in the online session to open multiple communication ports, with each port being associated with a particular communication protocol. The network computers communicate with other network computers using a particular port and a particular protocol, which is specified in the data structure comprised of the table **500** shown in FIG. **5**. The ports can comprise network sockets through which the instances of the application **440** communicate over the network. The network computers preferably communicate the port/protocol information, as well as the other information contained in the index list, by periodically sending communication messages to one another over the network.

Preferably, all of the computers that are participating in the online session keep their own copy of the table **500** index list. It should be appreciated that the table **500** is merely exemplary and that the initiating host application **440** could keep track of client index numbers and session master ownership in other manners using a wide variety of data structure formats. Alternatively, the session computers can share one or more copies of the table.

The initiating application **440** can operate in various communication configurations depending on how the application **440** assigns ownership of the session master. In a first configuration, shown in FIG. **6**, the initiating application **440** has assigned ownership of a session master **600** to a single computer, such as the dedicated server computer **420**. Therefore, this computer **420** has responsibility for all of the tasks associated with the session master function as dictated by the initiating application **440**. Thus, the application **440** operates in a client-server communication configuration with respect to the functions of the session master, with the server computer **420** serving data to the client computers **410** relating to the session master responsibilities.

It should be appreciated that any of the computers that are participating in the online session of the application **440** could have ownership of one or more of the session master tasks, such as where one of the client computers **410** is shown owning a session master **600***a*, which is shown in FIG. **6** using phantom lines. This indicates that the client computer has been assigned and is performing one or more session master tasks, along with or in place of the nominal session master server computer **420**. That is, there can be several instances of a session master among the computers participating in an online session, with each instance of a session master being assigned specific responsibilities and each session master task being assigned to a different network computer, or multiple tasks being assigned to the same computer. For example, FIG. **6** shows a situation where there are two session masters **600** and **600***a*, each being assigned responsibility for certain functionality relating to the online session of the application **440**. The server computer **420** has certain responsibilities and the client computer **410** also has certain responsibilities, as determined by the application. This is an integrated server configuration, where the client computer **410** that owns the session master **600***a* is functioning as an integrated server. An "integrated server" refers to a situation in which all clients send information to a client that is designated as an integrated server, and where that integrated server propagates the information to the other clients. The client acting as the integrated server can also own one or more session master tasks.

In another scheme, shown in FIG. **7**, the application **440** has distributed ownership of the session master **600** among several computers. In the illustrated example, both of the client computers **410** share ownership of the session master **600**. In such a case, both of the computers could perform the functions associated with the session master so that the network computers in FIG. **7** are in a peer-to-peer configuration. Regardless of the particular communication configuration in which the application **440** operates, the online session of the application includes various network participants comprised of the network computers that are running the application and interacting pursuant to the online session. If one of the participants in the online session were to exit the session, then this may result in an interruption for the other participants that remain in the session. In accordance with one aspect of the invention, the application **440** is configured to handle situations where a participant exits an online session in order to minimize the interruption for remaining participants.

This is described in more detail with reference to the flow diagram shown in FIG. **8**, which describes a process of managing the exiting of a participant in an online session of the multi-user application **440**. In the first operation, represented by the flow diagram box numbered **810**, it is determined that a participant of the online session of the application has exited the online session. The determination that a participant has exited the online session can be made in a variety of manners. In one embodiment, the instance of the application **440** on a network computer participating in the online session periodically causes the network computer to broadcast an update message notifying the other network computers of its presence in the online session. If no update message is received from a particular network computer within a predetermined amount of time, then it is deemed that the network computer has exited the online session. During establishment of the online session, the instance of the application **440** on the computer that starts the online session may assign a particular computer, such as the session master computer, with the responsibility for making the determination that a participant has exited the online session.

In the next operation, represented by the flow diagram box numbered **820**, the instance of the application **440** in one of the computers causes a notification message to be broadcast to all of the participants in the online session notifying them that a participant (the "exited participant") has exited the session. Only one of the network computers, such as, for example, the session master computer, broadcasts the notification message to all of the participants. If the session master computer is the exited computer, then one of the other network computers broadcasts the message, such as the computer with the next consecutive index after the session master computer. The notification message preferably includes the index that was previously assigned to the network computer for the exited participant. In this manner, the other participants can identify the exited participant by consulting the index table that was discussed above with respect to FIG. **5**.

The next operation differs based upon whether the participant that exited the session was responsible for performing any session managerial functions that would affect the other participants of the online session, as represented by the decision box numbered **830**. The managerial functions include functions such as filtering communication messages, assigning identification indices, score-keeping, keeping track of session times, keeping track of items that are located in an online world, keeping track of participant locations in an online world, etc. If the participant that exited the session was responsible for performing any such functions, a "Yes" result from the decision box **830**, then the process proceeds to the operation represented by the flow diagram box numbered **840**. In this operation, the application **440** reassigns the managerial responsibilities of the exited participant to a network computer of another participant that is still present in the online session. The reassignment of managerial responsibilities is preferably performed by an instance of the application **440** on a specific computer, such as the computer that has the next consecutive index after the index of the computer that exited the online session. For example, a computer with a first index may exit the session. The instance of the application on computer with the next consecutive index (as specified in the table **500** shown in FIG. **5**) then performs the re-assignment of the exited computer's duties.

The manner in which the application **440** re-assigns the responsibilities can vary. In one embodiment, the application **440** automatically selects the participant that is re-assigned the responsibilities based on certain factors, some of which are related to the conditions of the network computers of the participants. The conditions can include, for example, the communication environment, geographical location, and hardware specification of the network computers, as well as user-specified preferences.

The communication environment relates to whether the network computer of the participant has large bandwidth capabilities, such as through a cable-modem or DSL. Preferably, those participants with higher bandwidth capabilities are given higher preference for assuming the responsibilities of the exited participant. The geographic location of the participants can also be a factor in deciding which participant is re-assigned the responsibilities of the exited participant. For example, a participant that is centrally located to the other participants may be given a higher priority in order to minimize the latency for communications. The hardware specifications of the network computers that are participating in the online session are also a factor. The application **440** may give higher priority to network computers that have hardware capabilities that are most highly suited for the responsibilities that are being re-assigned, such as computers with powerful data processing capabilities.

In another embodiment of the operations of flow diagram box **840**, the application **440** simply randomly re-assigns the responsibilities of the exited participant to another of the participants of the online session. The application **440** can also consider user-specified preferences. Some users may specify to the application **440** that they do not want to be assigned the responsibility of managing any application functionality. The users may also specify that a particular participant should be re-assigned responsibilities should another participant exit the online session. Alternately, the application **440** may cause a message to be broadcast to all of the participants of the online session asking whether any of the participants would like to take over the responsibilities that were previously assigned to the exited participant.

The next operation is represented by the flow diagram box numbered **850**. This operation occurs after the application **440** has re-assigned the responsibilities of the exited participant. The operation of flow diagram box **850** also occurs if the exited participant did not have any responsibilities that needed re-assignment, which would have resulted in a "No" outcome from the decision box numbered **830**. In this operation, the application **440** attempts to obtain a new participant to replace the exited participant. The attempt is preferably performed by the instance of the application **440** on a specific computer, such as the session master computer. It should be appreciated that this operation differs from the operation of flow diagram box **840** in that this operation relates to obtaining a replacement participant for the online session to take the place of the exited participant, rather than re-assigning the managerial functions of the exited participant.

For example, the online session may be an online football game, where the participants are each a player on a common team. One of the participants may have been assigned managerial functions comprised of keeping track of the score and of the game time. That same participant may have played the role of the quarterback in the game. If the participant exits the online game during the game, then the application **440** re-assigns the managerial functions (i.e., score keeping and game time responsibilities) of the exited participant to another participant in the operation **840** and then, in operation **850**, attempts to obtain a new participant to replace the exited participant's role as quarterback.

The manner in which the application **440** attempts to obtain a replacement participant for the online session may vary. In

one embodiment, the application **440** automatically assigns a network computer, rather than a human, as a replacement for the exited participant. The network computer thereby would perform the functions of the exited participant. In another embodiment, the application **440** maintains a list of network computer that might be able to participate in the online session and then sends a message to those computers inviting them to participate in the session. The application **440** may put the online session in a pause mode while a replacement participant is obtained.

The application **440** is preferably developed using a software development kit (SDK) that provides a library of object and communication message definitions that are used in the application **440**. The software development kit includes an application interface through which applications that are developed using the SDK can run on a network system, such as the network system **400**. The application interface can reside in a central network server, such as the server **420**, to which network computers that have the application **440** can log onto in order to operate an online session of the application. By using the object and message types provided by the SDK, the application **440** can be developed to include the features described above. The SDK preferably includes an object definition structure that provides a client-based definition of objects that are utilized by the application **440**. The object definition includes a plurality of characteristics associated with each object and utilized by the application to effect interaction with clients over the computer network.

Once the application **440** has been developed using the SDK, the application **440** can be loaded onto one or more network computers and an online session can be established according to the operations shown in the flow diagram box of FIG. **9**. In the first operation, represented by the flow diagram box numbered **910**, a network computer on which the application **440** is loaded connects to a network computer that includes in memory that application interface software. For example, one or more of the client computers **410** of the network system **400** shown in FIG. **4** may have the application **440** loaded in memory and the server computer **420** may include the application interface. In such a case, the client computers **410** establish a communication connection with the server computer **410** over the network **430**.

In the next operation, represented by the flow diagram box numbered **920**, the application **440** registers objects according to the object definitions that are available in the library of the application interface. The application **440** also registers any message filters that will be utilized during the online session, as represented by the flow diagram box numbered **930**.

In the next operation, represented by the flow diagram box numbered **940**, the application **440** defines the session master and assigns ownership of the session master to one of the network computers. The ownership of the session master can be assigned to one computer or can be assigned to plural computers. The application **440** also specifies whether the ownership of the session master is dedicated to a particular computer or whether ownership can migrate to other computers.

During this operation, the application **440** assigns client indices to each of the network computers that will participate in the online session and also establishes the index table described above. The application **440** can be configured such that the first network computer to log onto the server computer will be the session master and also receive an initial index, such as an index of one or zero. It should be appreciated that the initial index may vary. Subsequent network computers to log on will then receive the next available index. After the ownership of the session master or session masters has been established, the online session of the application **440** is commenced, as represented by the flow diagram box numbered **950**.

As noted above, the network computers shown in the block diagram of FIG. **4** comprise nodes of a computer network system **400**. FIG. **10** is a block diagram of a computer in the system **400** of FIG. **4**, illustrating the hardware components included in one of the computers. Those skilled in the art will appreciate that the devices **410** and **420** may all have a similar computer construction, or may have alternative constructions consistent with the capabilities described herein.

FIG. **10** shows an exemplary computer **1000** such as might comprise any of the network computers. Each computer **1000** operates under control of a central processor unit (CPU) **1002**, such as a "Pentium" microprocessor and associated integrated circuit chips, available from Intel Corporation of Santa Clara, Calif., USA. A computer user can input commands and data from a keyboard and computer mouse **1004**, and can view inputs and computer output at a display **1006**. The display is typically a video monitor or flat panel display. The computer **1000** also includes a direct access storage device (DASD) **1008**, such as a hard disk drive. The memory **1010** typically comprises volatile semiconductor random access memory (RAM). Each computer preferably includes a program product reader **1012** that accepts a program product storage device **1014**, from which the program product reader can read data (and to which it can optionally write data). The program product reader can comprise, for example, a disk drive, and the program product storage device can comprise removable storage media such as a magnetic floppy disk, a CD-R disc, a CD-RW disc, or DVD disc.

Each computer **1000** can communicate with the others over a computer network **1020** (such as the Internet or an intranet) through a network interface **1018** that enables communication over a connection **1022** between the network **1020** and the computer. The network interface **1018** typically comprises, for example, a Network Interface Card (NIC) or a modem that permits communications over a variety of networks.

The CPU **1002** operates under control of programming steps that are temporarily stored in the memory **1010** of the computer **1000**. When the programming steps are executed, the computer performs its functions. Thus, the programming steps implement the functionality of the application **440**. The programming steps can be received from the DASD **1008**, through the program product storage device **1014**, or through the network connection **1022**. The program product storage drive **1012** can receive a program product **1014**, read programming steps recorded thereon, and transfer the programming steps into the memory **1010** for execution by the CPU **1002**. As noted above, the program product storage device can comprise any one of multiple removable media having recorded computer-readable instructions, including magnetic floppy disks and CD-ROM storage discs. Other suitable program product storage devices can include magnetic tape and semiconductor memory chips. In this way, the processing steps necessary for operation in accordance with the invention can be embodied on a program product.

Alternatively, the program steps can be received into the operating memory **1010** over the network **1020**. In the network method, the computer receives data including program steps into the memory **1010** through the network interface **1018** after network communication has been established over the network connection **1022** by well-known methods that will be understood by those skilled in the art without further explanation. The program steps are then executed by the CPU **1002** thereby comprising a computer process.

It should be understood that all of the network computers of the network system **400** illustrated in FIG. **4** may have a construction similar to that shown in FIG. **10**, so that details described with respect to the FIG. **10** computer **1000** will be understood to apply to all computers of the system **400**. It should be appreciated that any of the network computers can have an alternative construction, so long as the computer can communicate with the other computers over a network as illustrated in FIG. **4** and can support the functionality described herein.

For example, with reference to FIG. **11**, the client computers **420** can comprise a computer entertainment system, such as a video game system **1100**. FIG. **11** is a block diagram of an exemplary hardware configuration of the video game system **1100**.

The video game system **1100** includes a central processing unit (CPU) **1100** that is associated with a main memory **1105**. The CPU **1100** operates under control of programming steps that are stored in the OS-ROM **1160** or transferred from a game program storage medium to the main memory **1105**. The CPU **1100** is configured to process information and to execute instructions in accordance with the programming steps.

The CPU **1100** is communicatively coupled to an input/output processor (IOP) **1120** via a dedicated bus **1125**. The IOP **1120** couples the CPU **1100** to an OS ROM **1160** comprised of a non-volatile memory that stores program instructions, such as an operating system. The instructions are preferably transferred to the CPU via the IOP **1120** at start-up of the main unit **1100**.

The CPU **1100** is communicatively coupled to a graphics processing unit (GPU) **1110** via a dedicated bus **1115**. The GPU **1110** is a drawing processor that is configured to perform drawing processes and formulate images in accordance with instructions received from the CPU **1100**. For example, the GPU **1110** may render a graphics image based on display lists that are generated by and received from the CPU **1100**. The GPU may include a buffer for storing graphics data. The GPU **1110** outputs images to an audio-visual output device.

The IOP **1120** controls the exchange of data among the CPU **1100** and a plurality of peripheral components in accordance with instructions that are stored in an IOP memory **1130**. The peripheral components may include one or more input controllers **1122**, a memory card **1140**, a USB **1145**, and an IEEE 1394 serial bus **1150**. Additionally, a bus **1155** is communicatively coupled to the IOP **1120**. The bus **1155** is linked to several additional components, including the OS ROM **1160**, a sound processor unit (SPU) **1165**, an optical disc control unit **1175**, and a hard disk drive (HDD) **1180**.

The SPU **1165** is configured to generate sounds, such as music, sound effects, and voices, in accordance with commands received from the CPU **1100** and the IOP **1120**. The SPU **1165** may include a sound buffer in which waveform data is stored. The SPU **1165** generates sound signals and transmits the signals to speakers.

The disc control unit **1175** is configured to control a program reader, which can comprise, for example, an optical disk drive that accepts removable storage media such as a magnetic floppy disk, an optical CD-ROM disc, a CD-R disc, a CD-RW disc, a DVD disk, or the like.

The memory card **1140** may comprise a storage medium to which the CPU **1100** may write and store data. Preferably, the memory card **1140** can be inserted and removed from the IOP **1120**. A user can store or save data using the memory card **1140**. In addition, the video game system **1100** is preferably provided with at least one hard disk drive (HDD) **1180** to which data may be written and stored.

A data I/O interface, such as an IEEE 1394 serial bus **1150** or a universal serial bus (USB) **1145** interface, is preferably communicatively coupled to the IOP **1120** in order to allow data to be transferred into and out of the video game system **1100**, such as to the network **430** of FIG. **4**.

The system and method described above improve on the situation where a network user of an application, such as a game player, is performing as an Integrated Server (IS) for the application, thereby maintaining an application environment, so that the application would end for a conventional implementation of the application when that IS user wishes to log off. As described above, some applications (such as multi-user gaming applications) alternatively permit the functions (and data) of the departing user to be migrated from the departing user to a different user, who will continue with the online session and will take over the duties of the IS. This type of hand-off is typically rather cumbersome and might not be accomplished smoothly. In the case of a gaming environment, for example, a departing player might abruptly disappear from the game environment, thereby disrupting the gaming experience of the other players. The multi-user application in accordance with the invention permits continued use of the application, even with users departing and joining, by notifying all user machines when another user has departed from the session. Suitable adjustments can be made for a more pleasing application environment. The notification occurs through a Disconnect function that ensures proper IS operation and communications. That is, an application server or IS of the application can broadcast a message to all clients in an application environment to notify them that a user has departed or has joined, and can ensure appropriate functionality, if necessary, of the users.

For a system with a network device operating as an Integrated Server (IS) as described above by serving application data to other users, the failure or departure of an IS from the application environment is responded to by assigning a different user as a new IS. The application can assign a new IS by automatically carrying out a replacement process, or by sending a broadcast message to all users and awaiting replies. For automatic selection, the application can assign the new IS in accordance with considerations that include the bandwidth available to the user, the geographic location of the potential new IS, a user's indicated preference for consideration as an IS, the technical specification and resources available at the user's machine, or through a random selection process. If the application is designed so that it sends a broadcast message, then the message will typically solicit voluntary agreement to operate as the new IS.

In addition, a new user who logs in to the system after an IS failure and who otherwise might have been registered with the failed IS can instead be diverted to a different IS, reducing the workload of the group being served by the now-unavailable IS. In this way, newcomers who wish to join the application environment of an IS can instead be moved to a different IS and different user group. Alternatively, the application can respond to a failed IS by dissolving or disbanding the online session of the group administered by the unavailable IS and forming a new online group with a new IS. These alternatives can be selected by an application developer who is configuring an application for operation in accordance with the present invention.

If an individual user leaves during an online session, the result can be somewhat more problematic. In the online gaming context, for example, a certain minimum number of users (players) are required for a game to proceed. In accordance with the invention, the application can respond by sending a message to other users on the network, inviting others to join the online session and participate in the multi-user application (such as a game). Alternatively, the application can be configured so as to invoke an Artificial Intelligence module to carry out the duties of the Integrated Server.

The present invention has been described above in terms of a presently preferred embodiment so that an understanding of the present invention can be conveyed. There are, however, many configurations for the system and application not specifically described herein but with which the present invention is applicable. The present invention should therefore not be seen as limited to the particular embodiment described herein, but rather, it should be understood that the present invention has wide applicability with respect to multi-user applications generally. All modifications, variations, or equivalent arrangements and implementations that are within the scope of the attached claims should therefore be considered within the scope of the invention.

What is claimed is:

1. A method of managing participants in an online session, comprising:

participating in an online session including a plurality of participants communicatively linked to one another via a gaming network, wherein each of the participants includes a computing device;

detecting that a first participant responsible for certain managerial functionality associated with the online session has disconnected from the online session, wherein detecting that the first participant has disconnected from the online sessions occurs when an update message periodically broadcast by the first participant is not received within a predetermined amount of time;

broadcasting a notification to the remaining participants from the plurality of participants in the online session that the first participant has disconnected from the online session, wherein the notification is broadcast following detection of the first participant having disconnected from the online session; and

reassigning the functionality associated with the first participant based on one or more determinations made by one or more of the remaining participants in the online session.

2. The method of claim **1**, wherein each of the participant computing devices are identified by an index identification number.

3. The method of claim **2**, wherein the broadcast indicating that the first participant has disconnected from the online session is initiated by the participant computing device with the next consecutive index identification number after the index identification number of the disconnected first participant.

4. The method of claim **1**, wherein the one or more determinations comprises a condition of the gaming network.

5. The method of claim **1**, wherein the one or more determinations comprises the geographical location of one or more of the participant network computers in the online session.

6. The method of claim **1**, wherein the one or more determinations comprises a hardware specification of one or more of the participant network computers in the online session.

7. The method of claim **1**, wherein the one or more determinations comprises a preference specified by a user of one of the participant network computers in the online session.

8. The method of claim **1**, further comprising accepting a new participant to the online session to replace the disconnected first participant.

9. The method of claim **8**, wherein the new participant is controlled by a user.

10. The method of claim **9**, wherein the new user-controlled participant joins a pre-existing team of participants in the online session.

11. The method of claim **8**, wherein the new participant is an automated participant not under the control of a user.

12. A system for managing participants in an online session, the system comprising:

a gaming network for establishing the online session amongst a plurality of participant computing devices, wherein each of the participant computing devices in the online session are communicatively linked to one another via the gaming network;

a first participant computing device from the plurality of participant computing devices, wherein the first participant computing device is responsible for certain managerial functionality associated with the online session, the first participant computing device configured to periodically broadcast an update message to the other participant computing devices in the online session; and

a second participant computing device configured to:

receive the periodically broadcast update message, wherein the second participant computing device will determine that the first participant computing device has disconnected from the online session if the update message is not received within a predetermined period of time,

broadcast a notification to the remaining participant computing devices from the plurality of participant computing devices in the online session that the first participant computing device has disconnected from the online session, wherein the notification is broadcast following the determination that the first participant computing device has disconnected from the online session, and

re-assign the functionality associated with the first participant computing device based on one or more determinations made by one or more of the remaining participant computing devices in the online session.

13. The system of claim **12**, wherein the one or more determinations comprises a condition of the gaming network.

14. The system of claim **12**, wherein the one or more determinations comprises the geographical location of one or more of the participant computing devices in the online session.

15. The system of claim **12**, wherein the one or more determinations comprises a hardware specification of one or more of the participant computing devices in the online session.

16. The system of claim **12**, wherein the one or more determinations comprises a preference specified by a user of one of the participant computing devices in the online session.

17. The system of claim **12**, wherein the re-assignment of functionality is random.

18. The system of claim **12**, wherein the second participant computing device is further configured to accept a new participant computing device to replace the disconnected participant computing device.

19. The system of claim **18**, wherein the new participant computing device is controlled by a user.

20. The system of claim **19**, wherein the new user-controlled participant computing device joins a team of participant computing devices in the online session.

21. The system of claim **18**, wherein the new participant computing device is an automated participant computing device not under the control of a user.

22. A computer-readable storage medium having embodied thereon a program, the program being executable by a computer to perform a method for managing participants in an online session, the method comprising:

detecting that a first participant responsible for certain managerial functionality associated with the online session has disconnected from the online session, wherein detecting that the first participant has disconnected from the online sessions occurs when an update message periodically broadcast by the first participant is not received within a predetermined amount of time;

broadcasting a notification to the remaining participants from the plurality of participants in the online session that the first participant has disconnected from the online session, wherein the notification is broadcast following detection of the first participant having disconnected from the online session; and

reassigning the functionality associated with the first participant based on one or more determinations made by one or more of the remaining participants in the online session.

23. The computer-readable storage medium of claim **22**, wherein the functionality is re-assigned to an existing participant in the online session.

24. The computer-readable storage medium of claim **22**, wherein the broadcast indicating that the first participant has disconnected from the online session is initiated by a participant computing device with a next consecutive index identification number after the index identification number of the disconnected first participant, each of the participant computing devices having been assigned an index identification number at the initiation of the online session.

25. The computer-readable storage medium of claim **22**, the method further comprising accepting a new participant to the online session to replace the disconnected first participant.

26. The computer-readable storage medium of claim **25**, wherein the new participant is controlled by a user.

27. The computer-readable storage medium of claim **25**, wherein the new participant is an automated participant not under the control of a user.

* * * * *

EXHIBIT 91

US007792968B2

(12) **United States Patent**
Datta et al.

(10) **Patent No.: US 7,792,968 B2**
(45) **Date of Patent: Sep. 7, 2010**

(54) METHOD OF MAINTAINING A PEER-TO-PEER RELAY NETWORK

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America LLC**, Foster City, 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.: **11/895,968**

(22) Filed: **Aug. 28, 2007**

(65) **Prior Publication Data**

US 2008/0046554 A1 Feb. 21, 2008

Related U.S. Application Data

(62) Division of application No. 10/700,798, filed on Nov. 3, 2003, now Pat. No. 7,627,678.

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/226**; 709/224; 709/227

(58) **Field of Classification Search** 709/223, 709/225, 227, 238, 224, 226, 243, 244
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,630,184	A *	5/1997	Roper et al.	709/237
5,701,427	A	12/1997	Lathrop	
6,421,347	B1 *	7/2002	Borgstahl et al.	370/310
6,701,344	B1	3/2004	Holt et al.	
7,065,579	B2 *	6/2006	Traversat et al.	709/225
7,130,921	B2 *	10/2006	Goodman et al.	709/238

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999

(Continued)

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

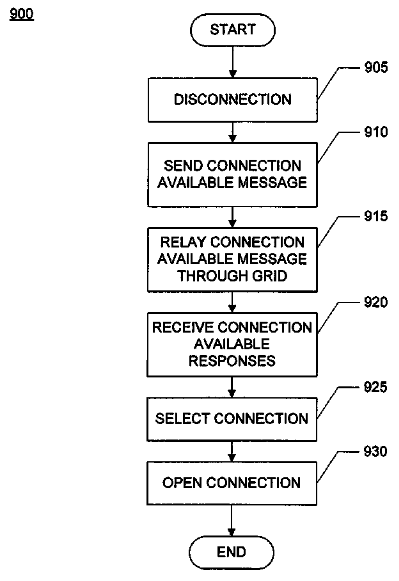
(Continued)

Primary Examiner—Ramy M Osman
(74) *Attorney, Agent, or Firm*—Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

A method of maintaining a peer-to-peer relay network includes sending a maintenance message from a peer computer system to each of one or more connected peer computer systems connected to the peer computer system and evaluating any responses received from the one or more connected peer computer systems. The connection between the peer computer system and a connected peer computer system is closed when the response from that connected peer computer system is not acceptable. Each peer computer system is connected to a number of other peer computer systems that is less than or equal to a connection limit and each peer computer system is configured to relay data to peer computer systems connected to that peer computer system according to a set of one or more relay rules.

7 Claims, 31 Drawing Sheets


900
START
905
DISCONNECTION
910
SEND CONNECTION AVAILABLE MESSAGE
915
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
920
RECEIVE CONNECTION AVAILABLE RESPONSES
925
SELECT CONNECTION
930
OPEN CONNECTION
END

U.S. PATENT DOCUMENTS

7,174,382	B2 *	2/2007	Ramanathan et al.	709/227
7,177,950	B2	2/2007	Narayan et al.	
7,263,070	B1 *	8/2007	Delker et al.	709/227
7,272,636	B2 *	9/2007	Pabla	709/223
2001/0044339	A1	11/2001	Cordero et al.	
2002/0006114	A1 *	1/2002	Bjelland et al.	370/355
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0107935	A1 *	8/2002	Lowery et al.	709/217
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1	12/2002	Traversat et al.	
2003/0055892	A1	3/2003	Huitema et al.	
2003/0104829	A1 *	6/2003	Alzoubi et al.	455/517
2004/0103179	A1	5/2004	Damm et al.	
2005/0007964	A1 *	1/2005	Falco et al.	370/256
2005/0063409	A1	3/2005	Oommen	
2005/0080858	A1	4/2005	Pessach	

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups" Proceedings of the 7TH ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. CONF. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

* cited by examiner

FIG. 1

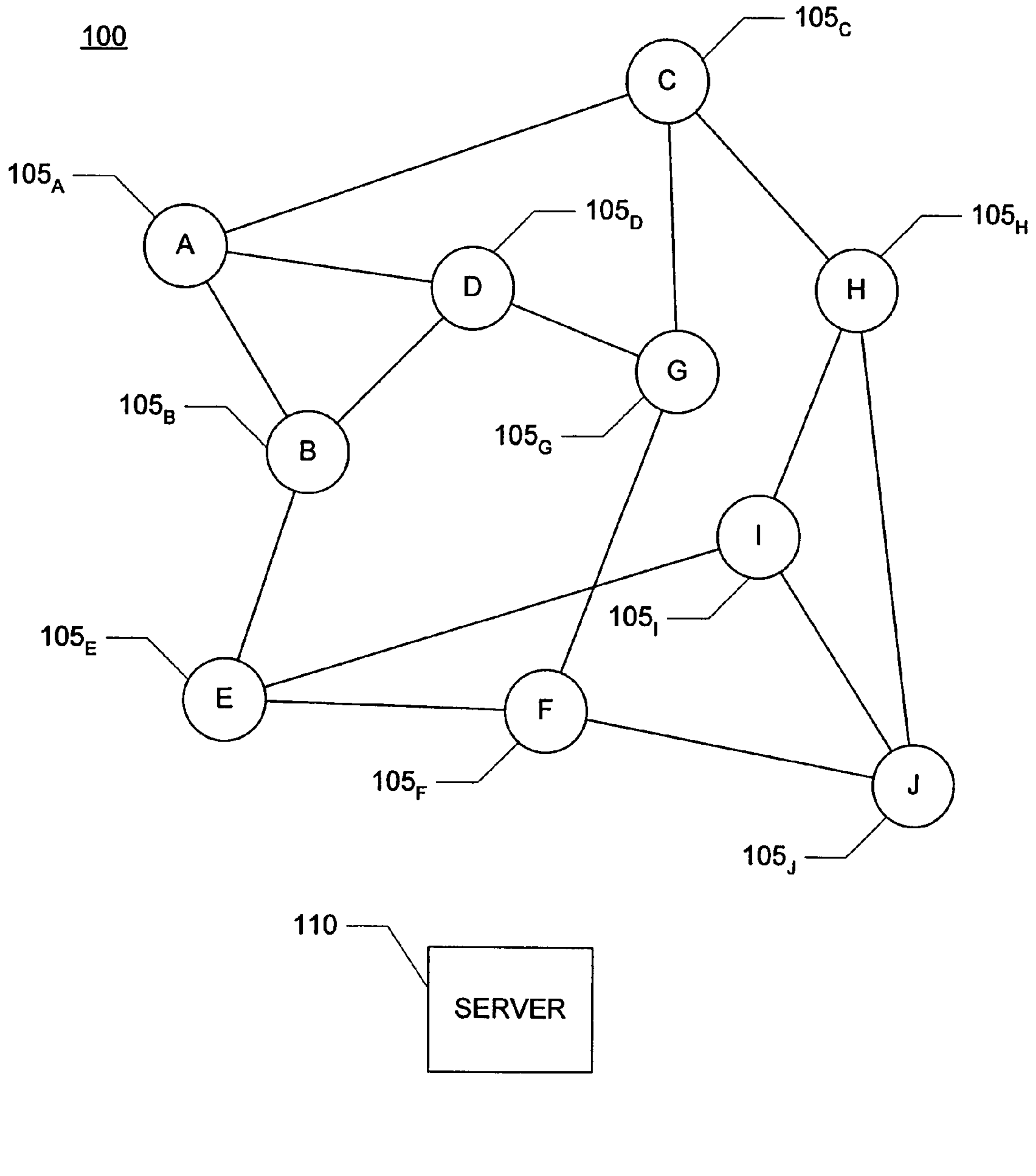

100
105_C
C
105_A
A
105_D
D
105_H
H
G
105_B
B
105_G
I
105_I
105_E
E
F
105_F
J
105_J
110
SERVER

FIG. 2

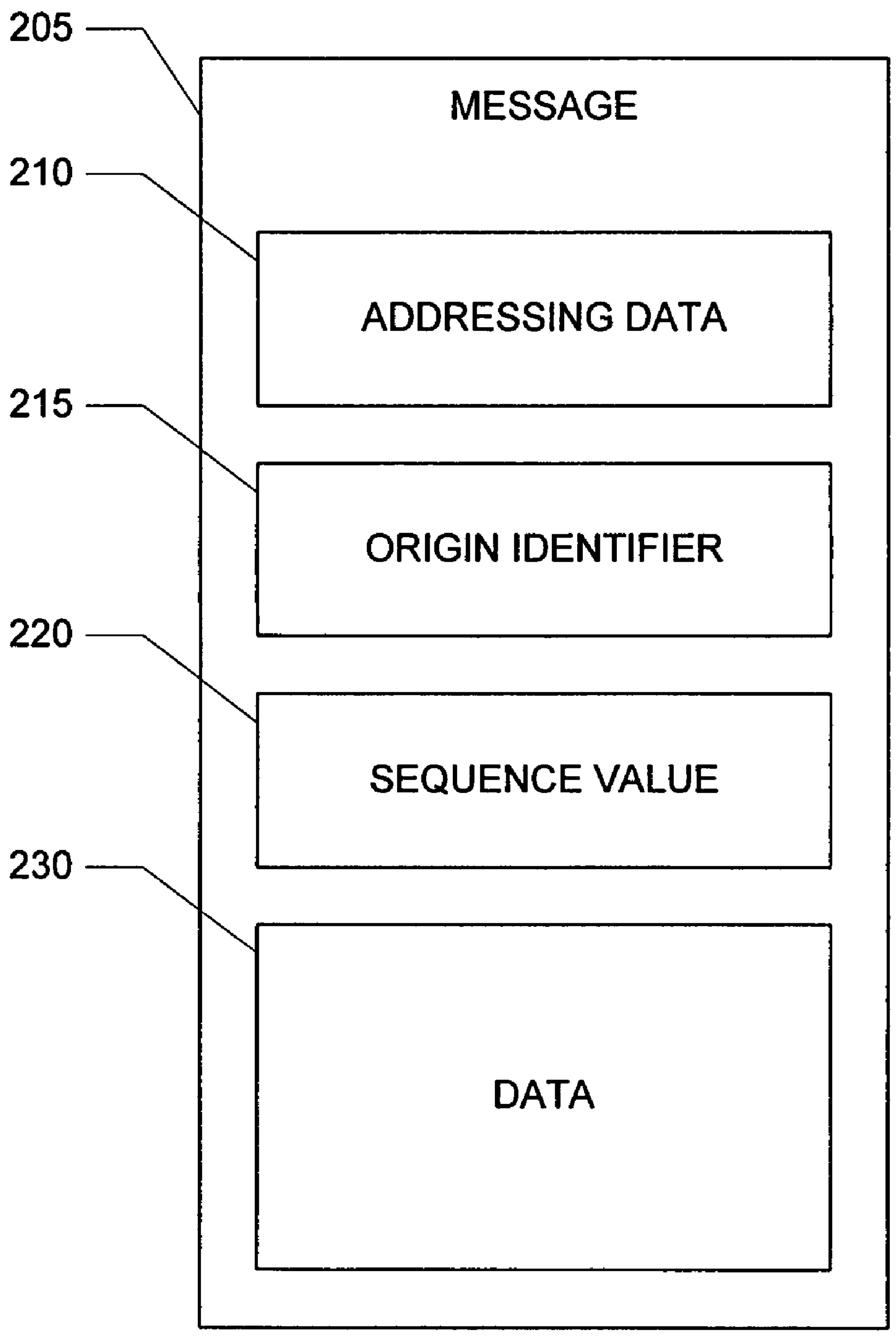

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

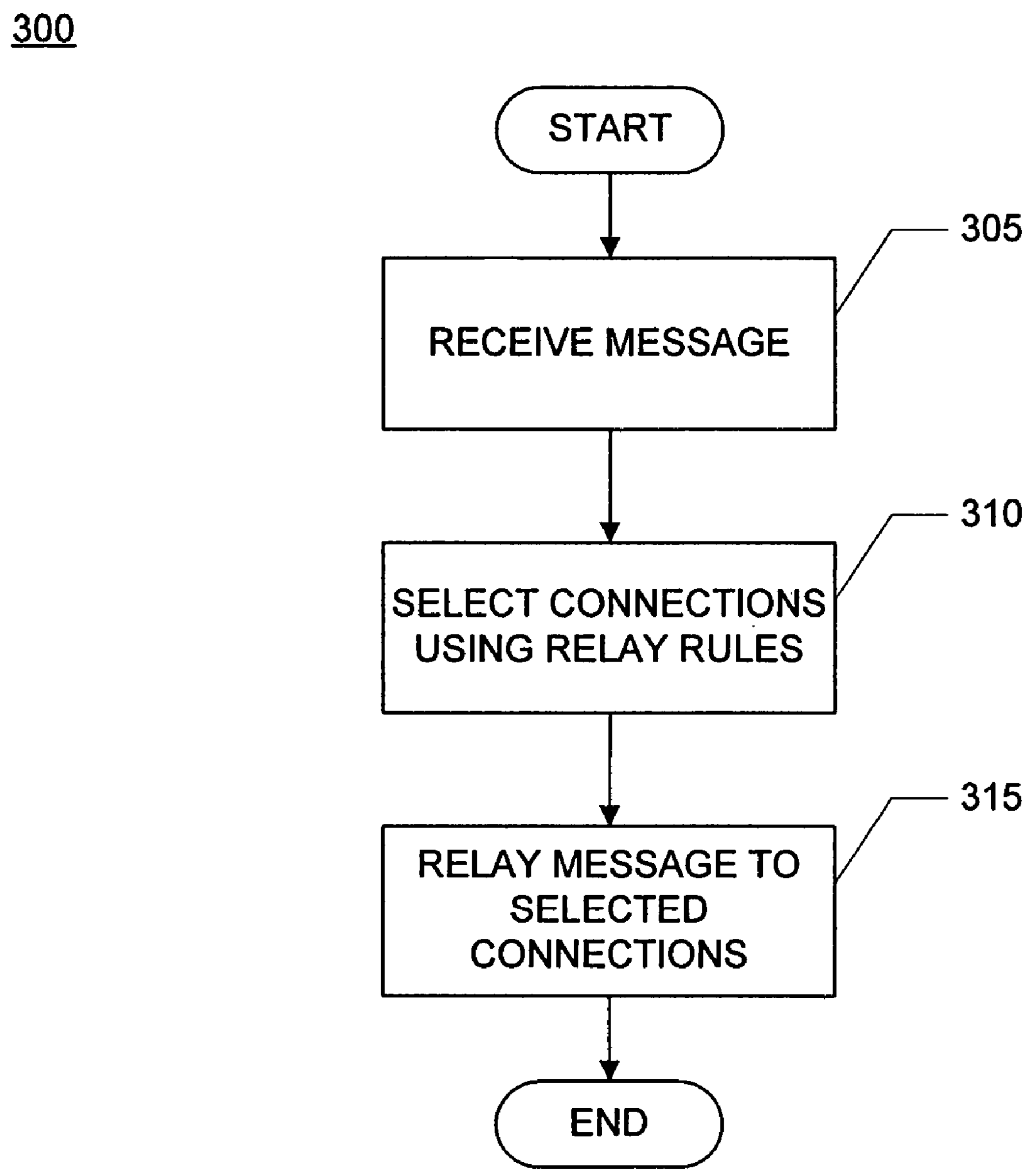

300
START
305
RECEIVE MESSAGE
310
SELECT CONNECTIONS USING RELAY RULES
315
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 4

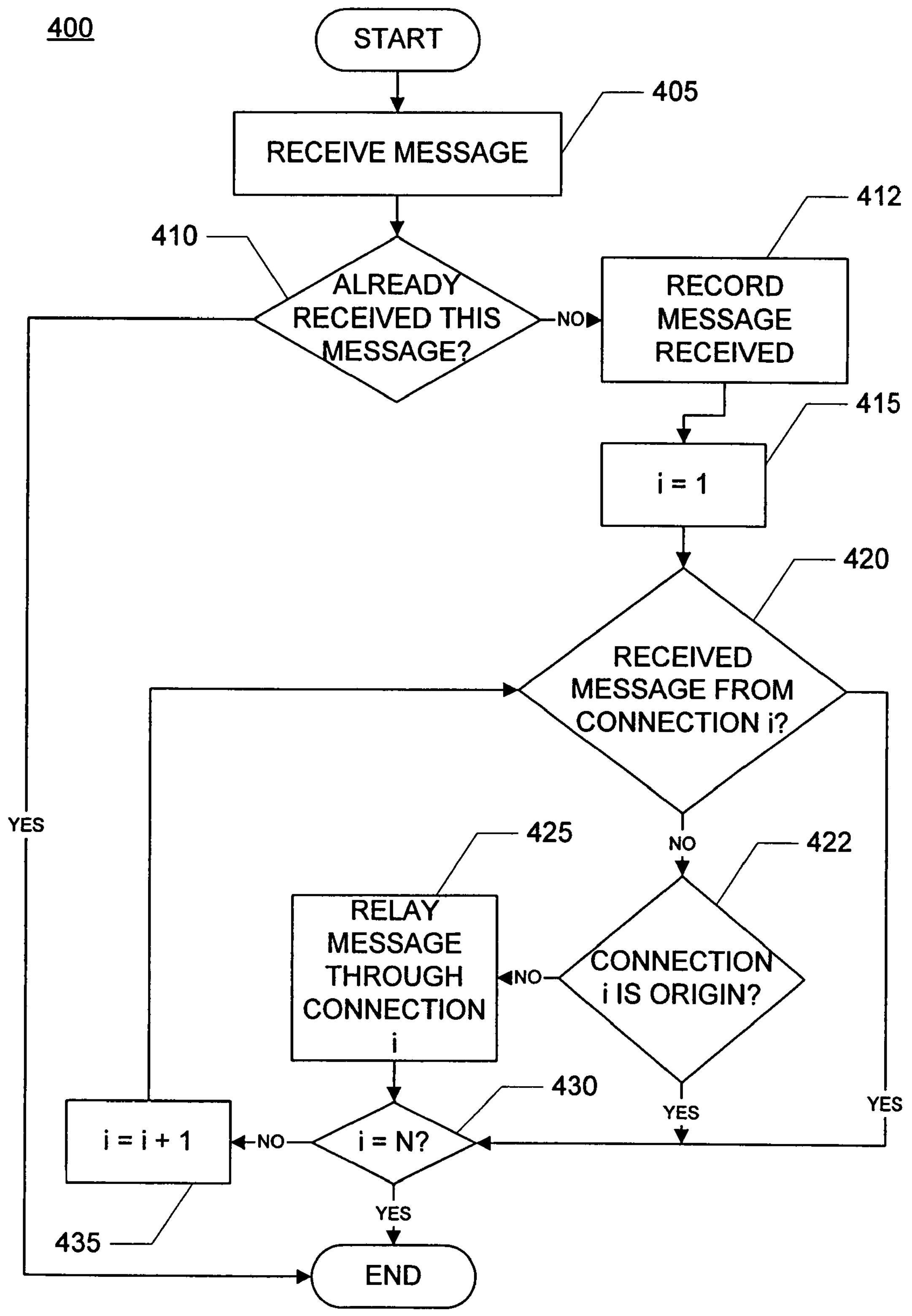

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
YES
RECORD MESSAGE RECEIVED
412
i = 1
415
RECEIVED MESSAGE FROM CONNECTION i?
420
NO
YES
CONNECTION i IS ORIGIN?
422
NO
YES
RELAY MESSAGE THROUGH CONNECTION i
425
i = N?
430
NO
YES
i = i + 1
435
END

FIG. 5

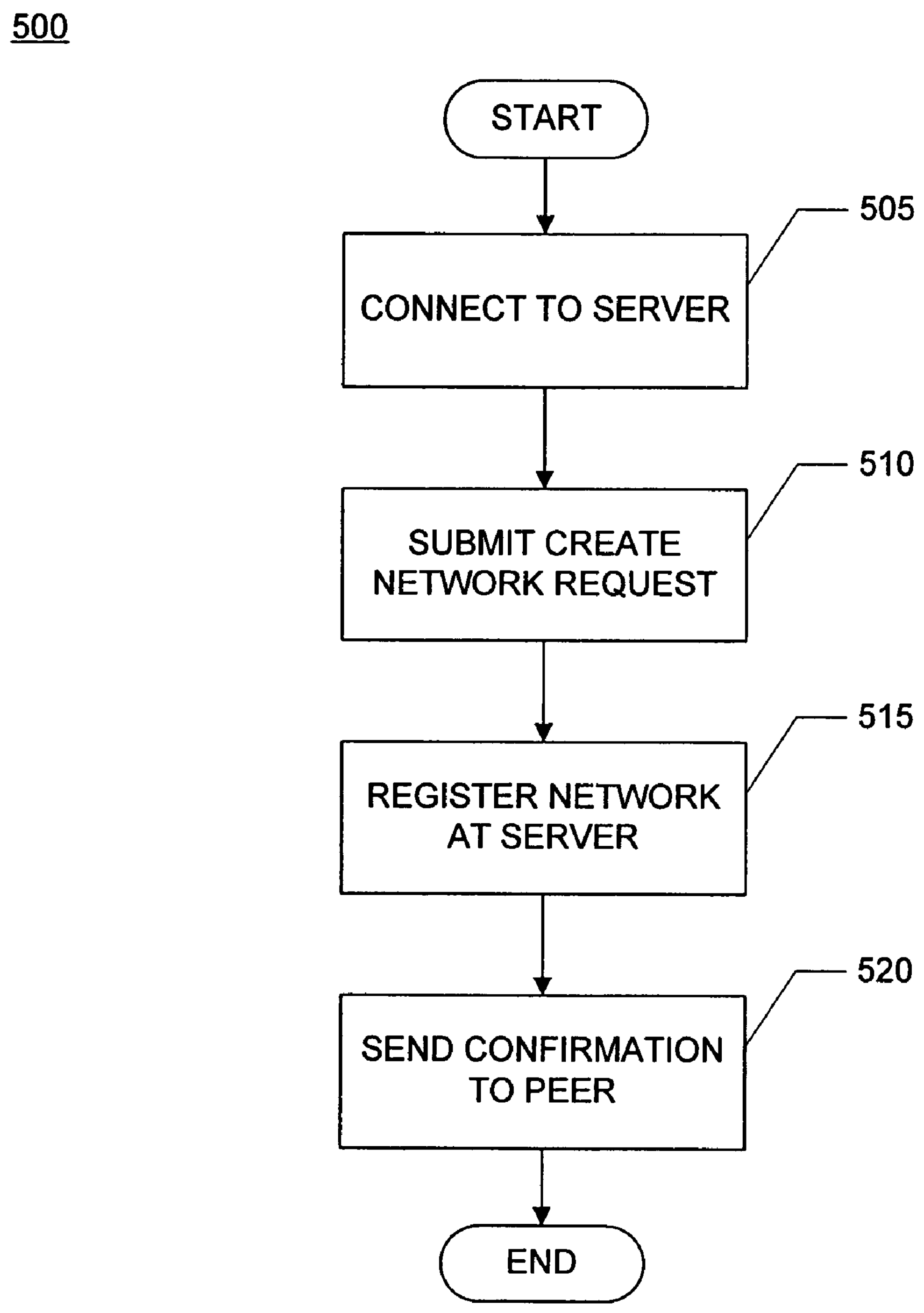

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

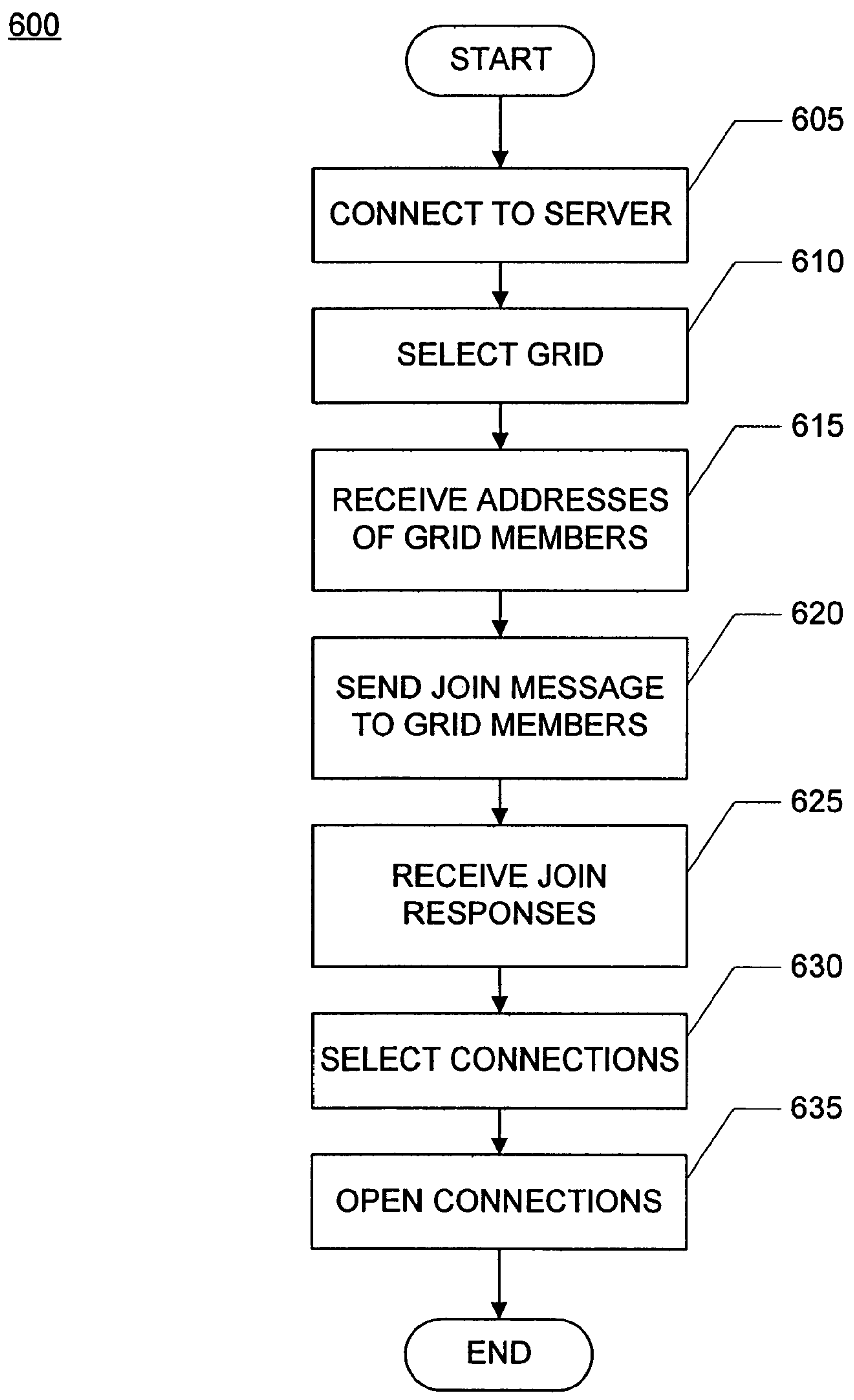

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

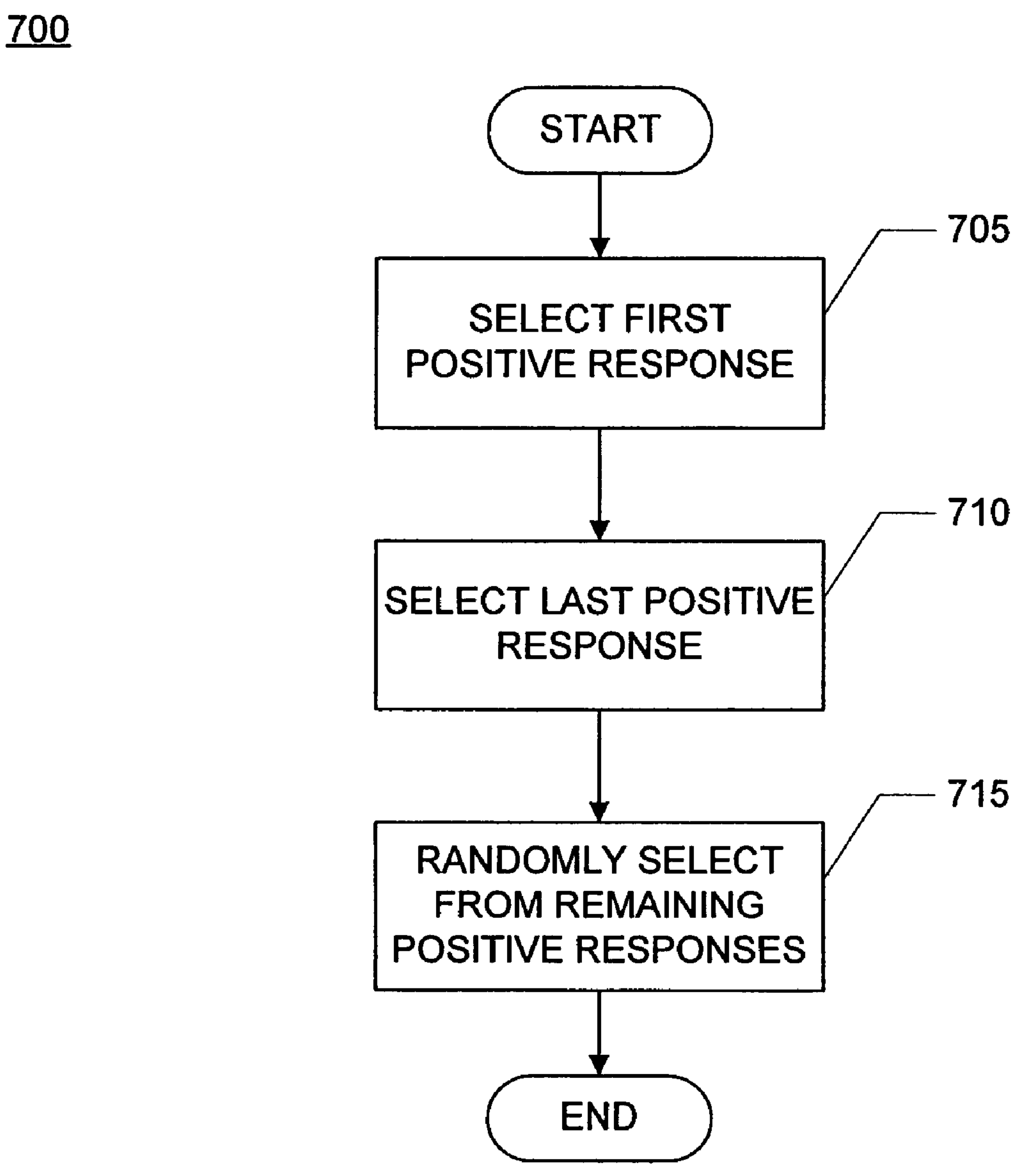

700
START
705
SELECT FIRST POSITIVE RESPONSE
710
SELECT LAST POSITIVE RESPONSE
715
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
END

FIG. 8

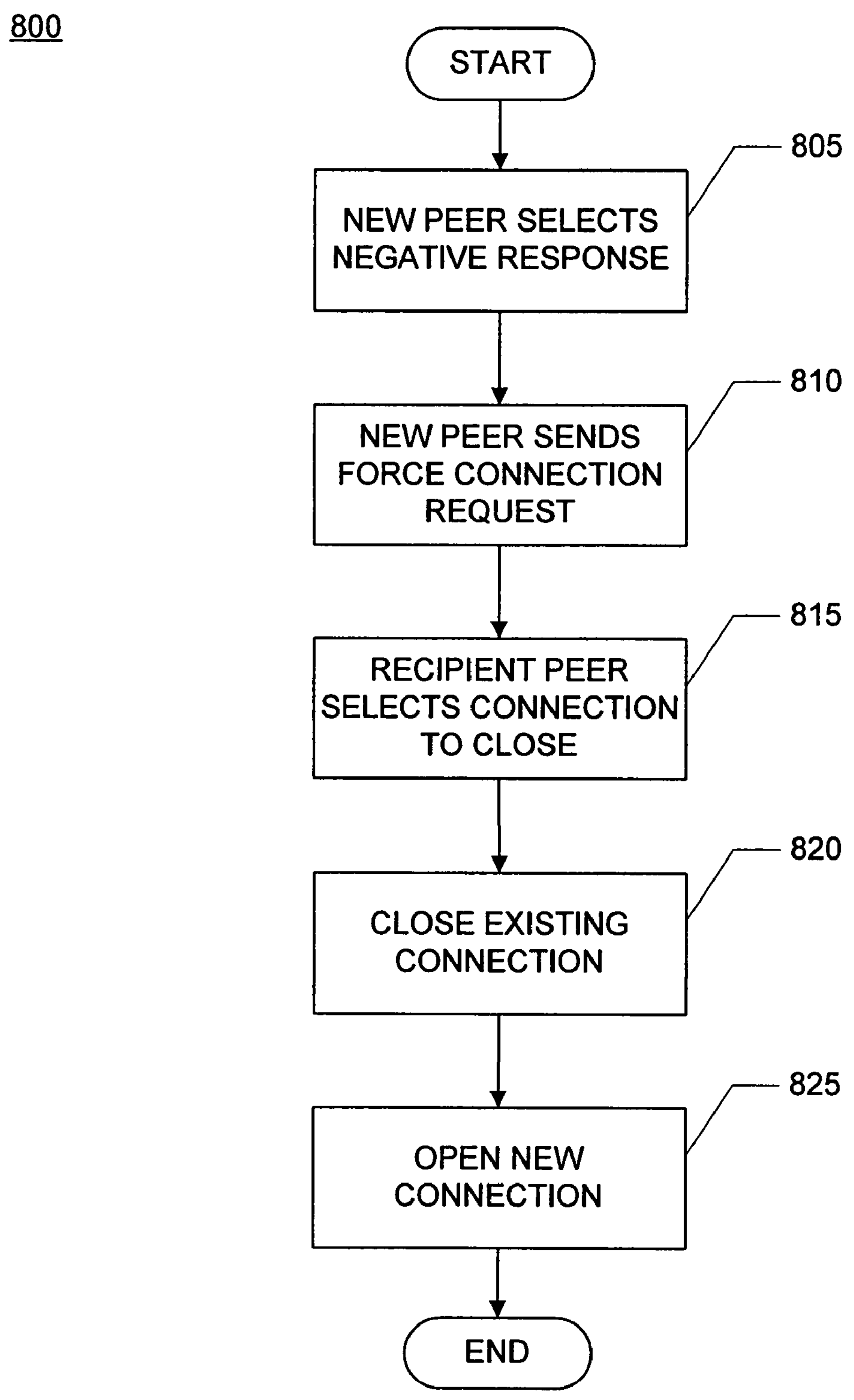

800
START
NEW PEER SELECTS NEGATIVE RESPONSE
805
NEW PEER SENDS FORCE CONNECTION REQUEST
810
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
815
CLOSE EXISTING CONNECTION
820
OPEN NEW CONNECTION
825
END

FIG. 9

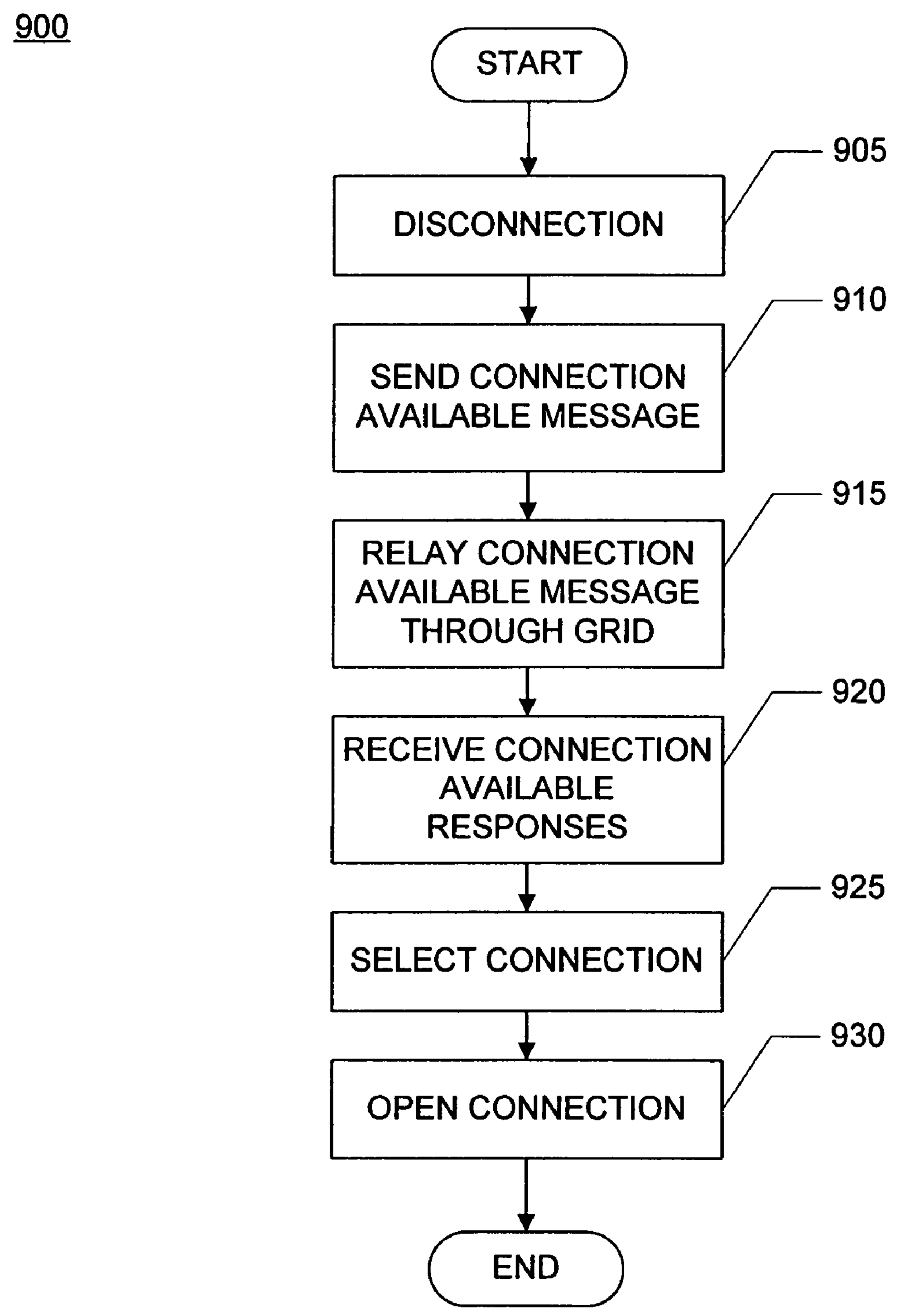

900
START
905
DISCONNECTION
910
SEND CONNECTION AVAILABLE MESSAGE
915
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
920
RECEIVE CONNECTION AVAILABLE RESPONSES
925
SELECT CONNECTION
930
OPEN CONNECTION
END

FIG. 10

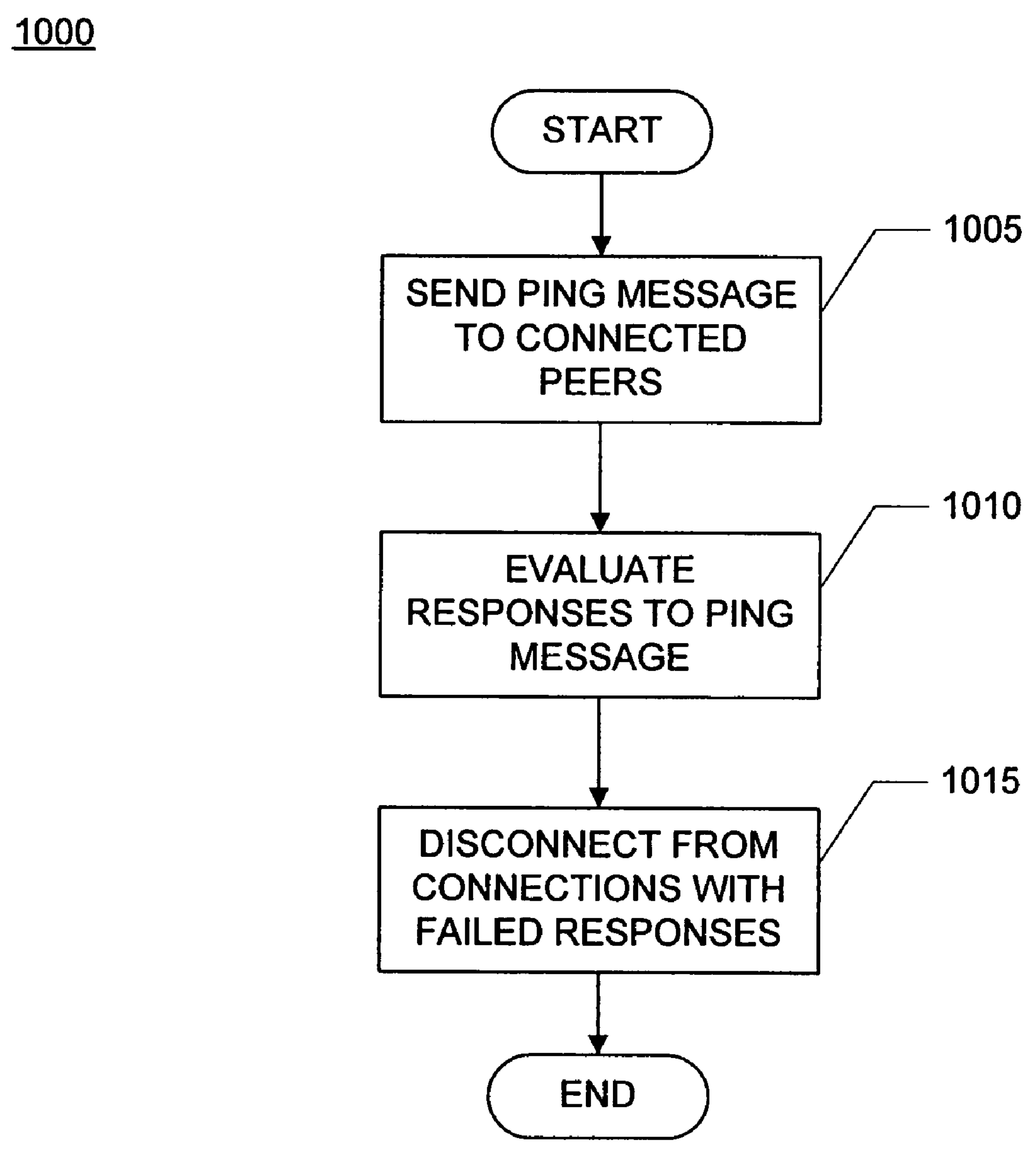

1000
START
1005
SEND PING MESSAGE TO CONNECTED PEERS
1010
EVALUATE RESPONSES TO PING MESSAGE
1015
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
END

FIG. 11

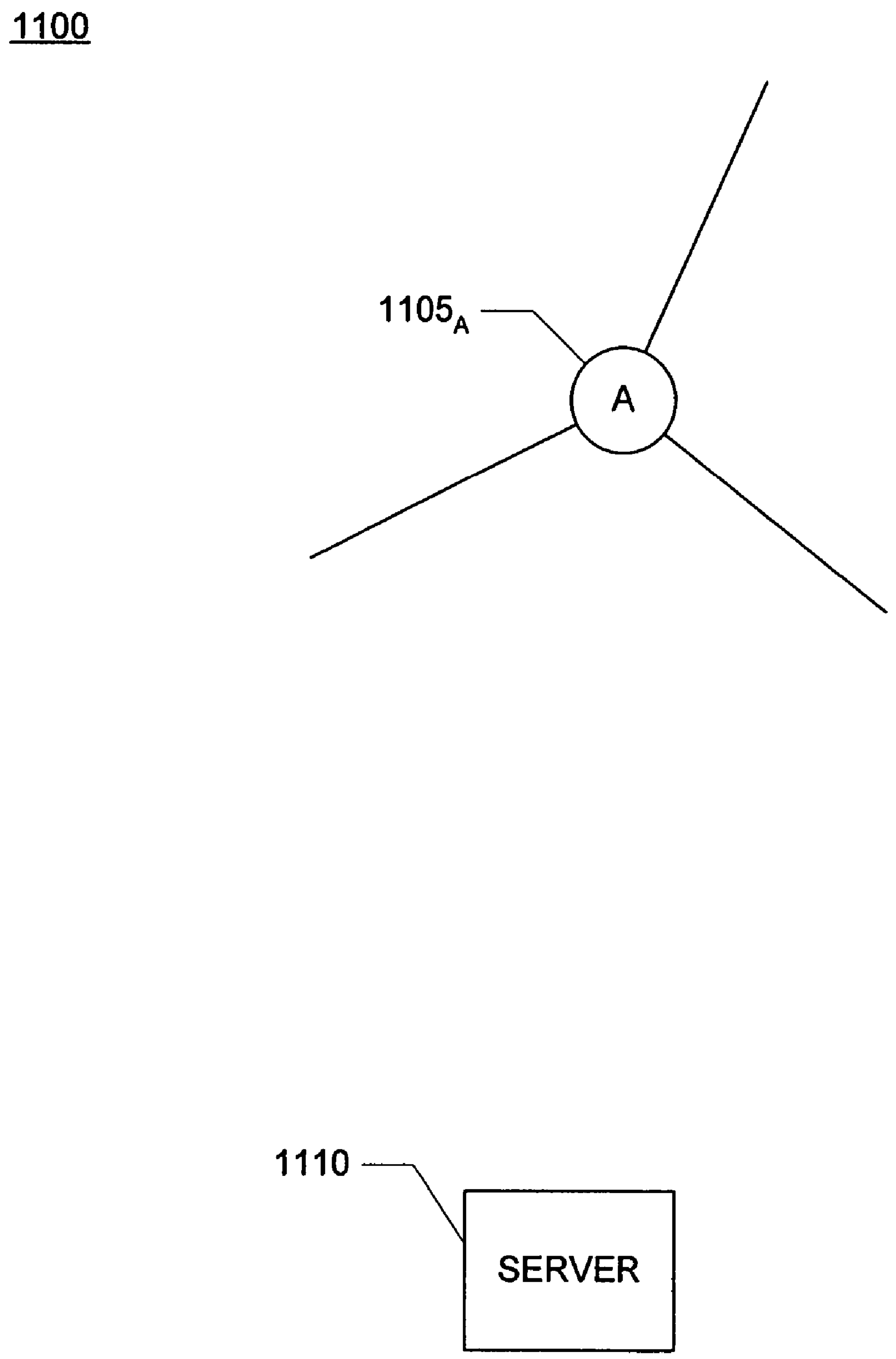


FIG. 12

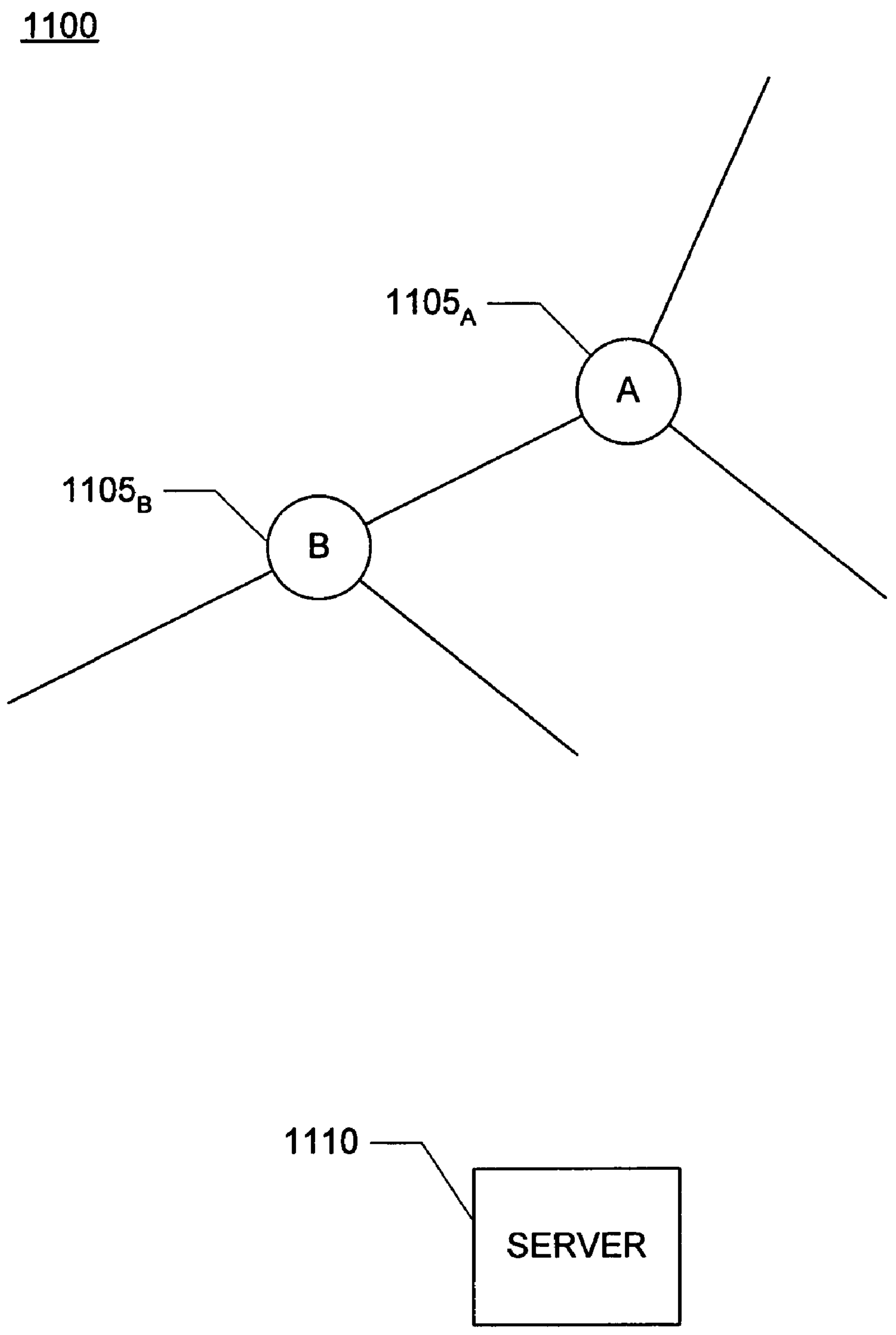


FIG. 13

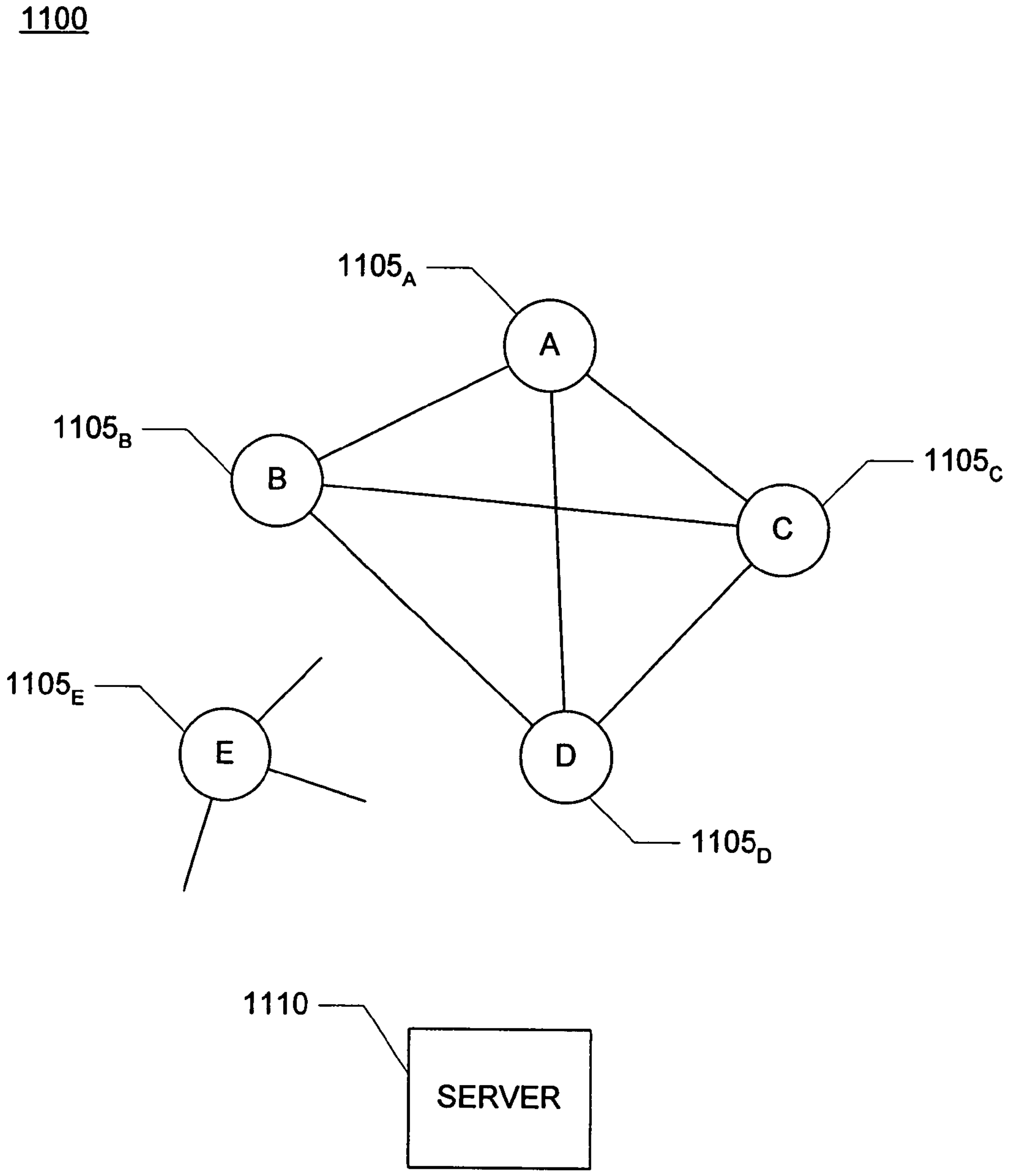


FIG. 14

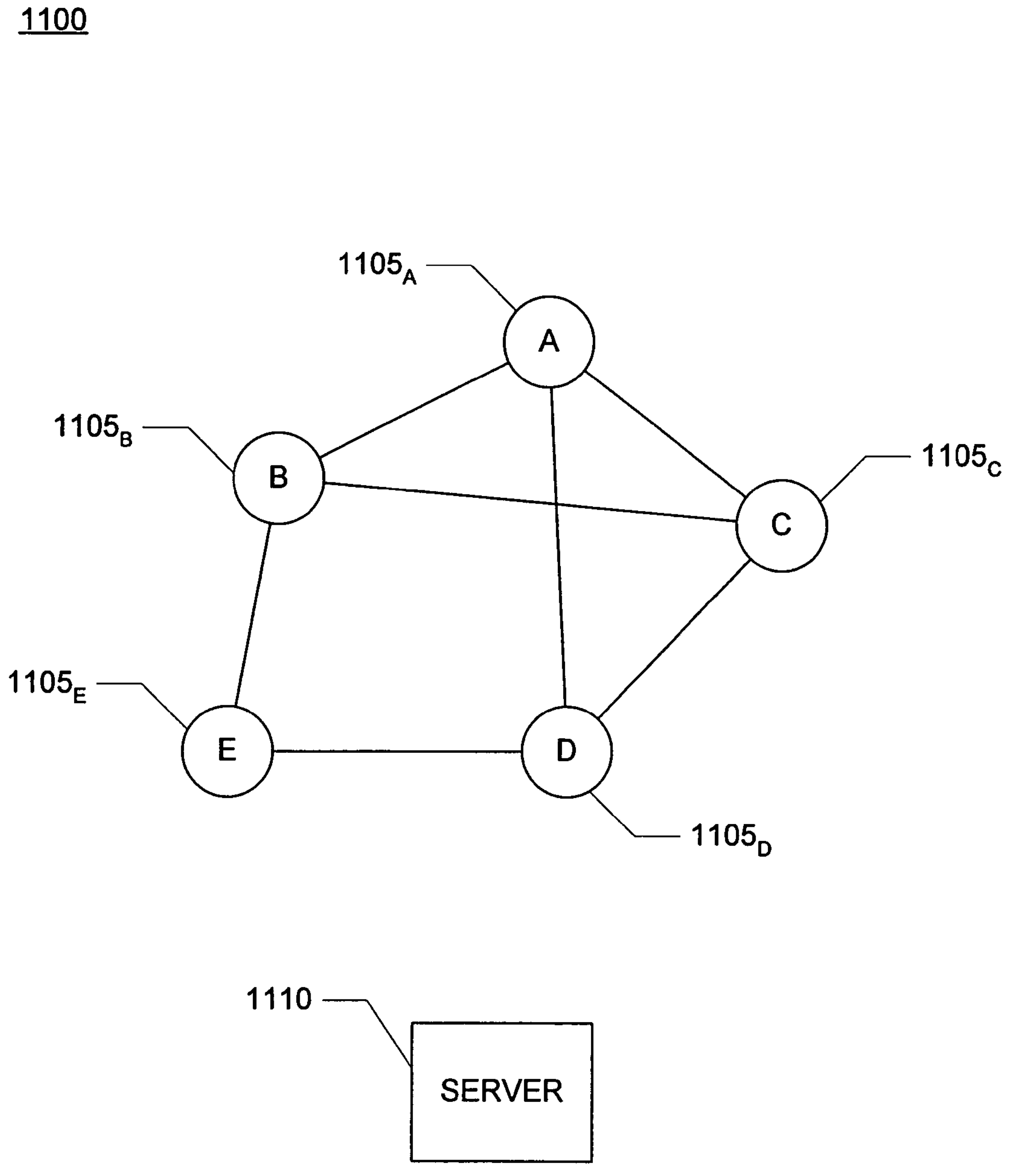


FIG. 15

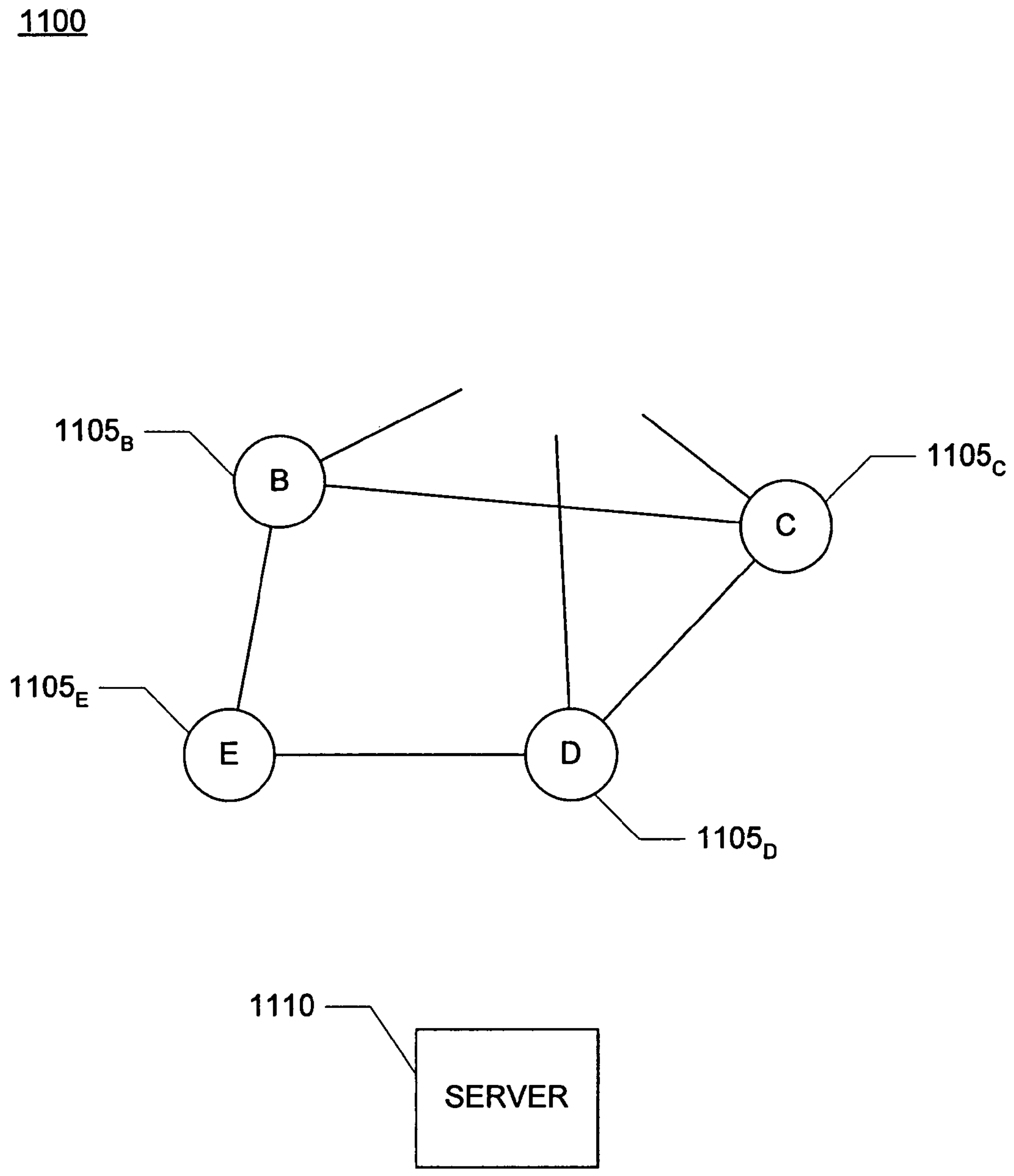


FIG. 16

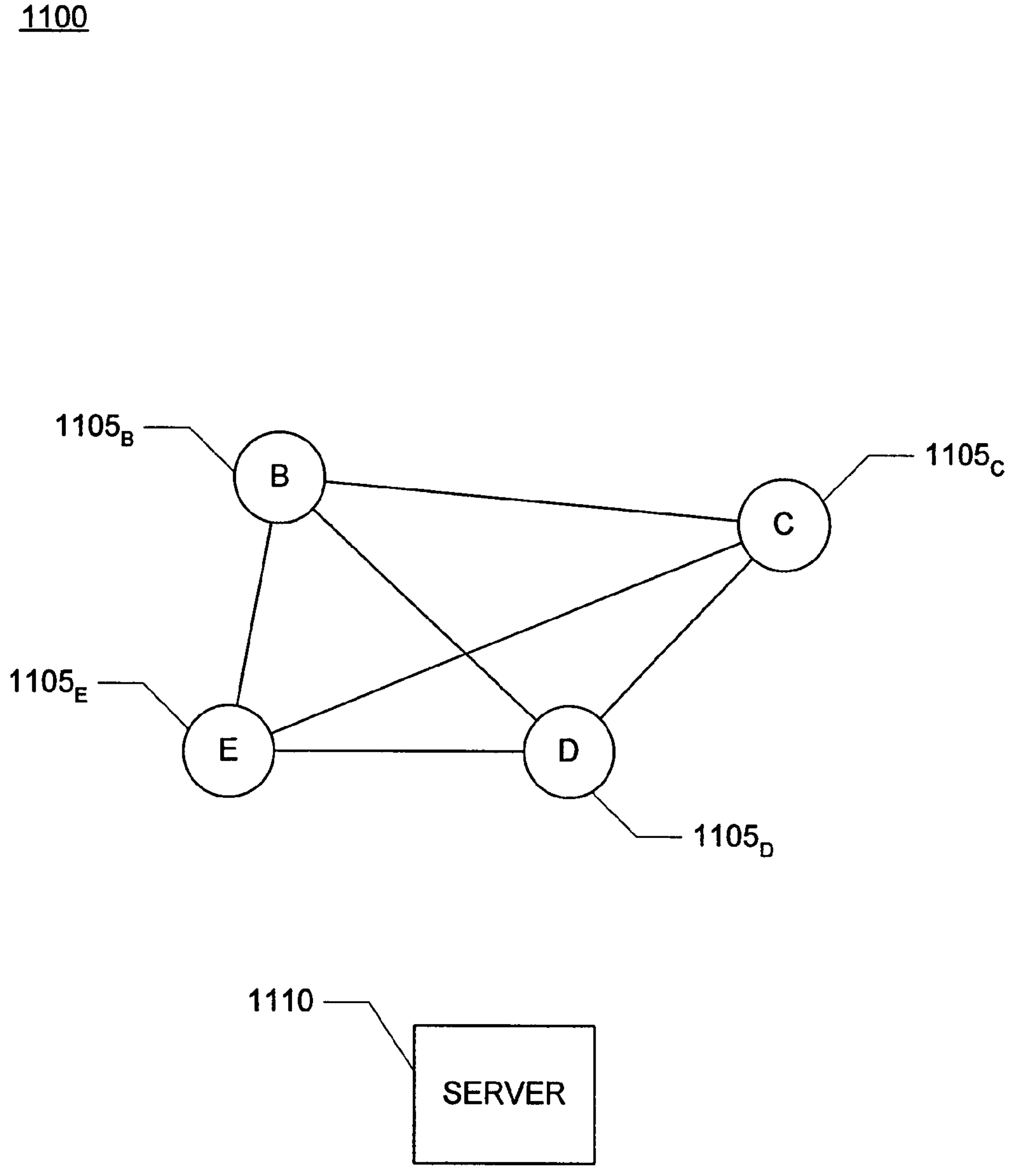


FIG. 17

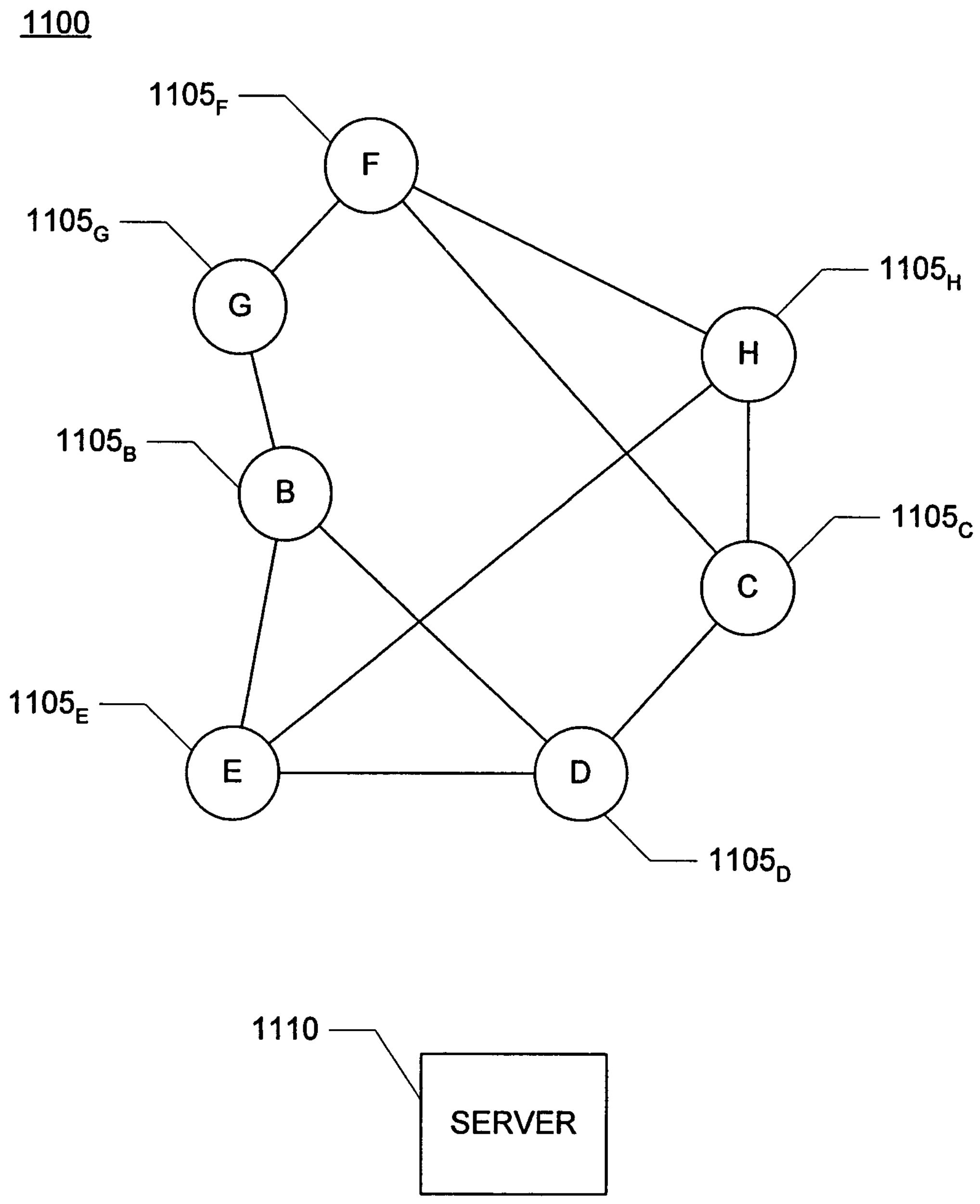

1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 18

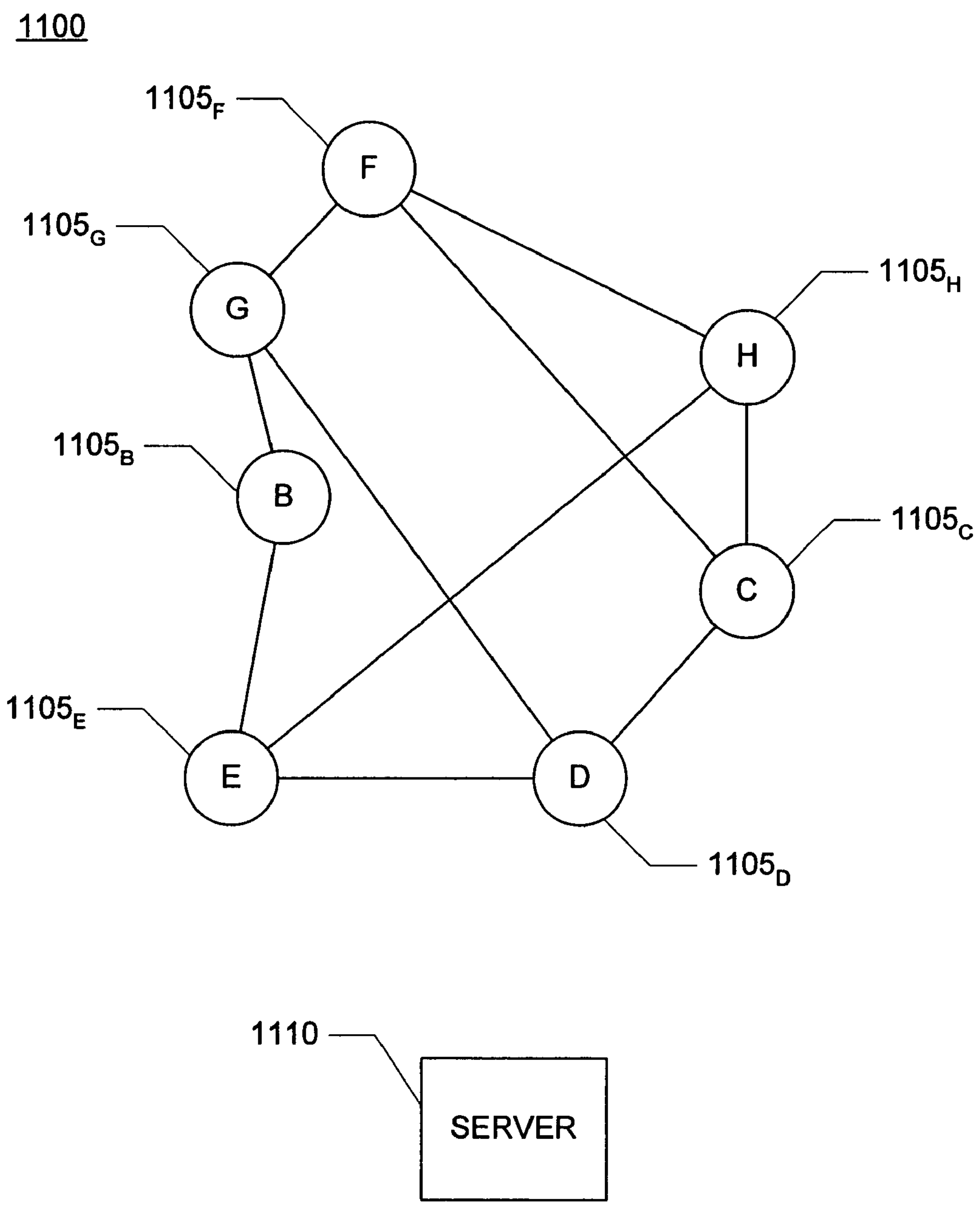

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 19

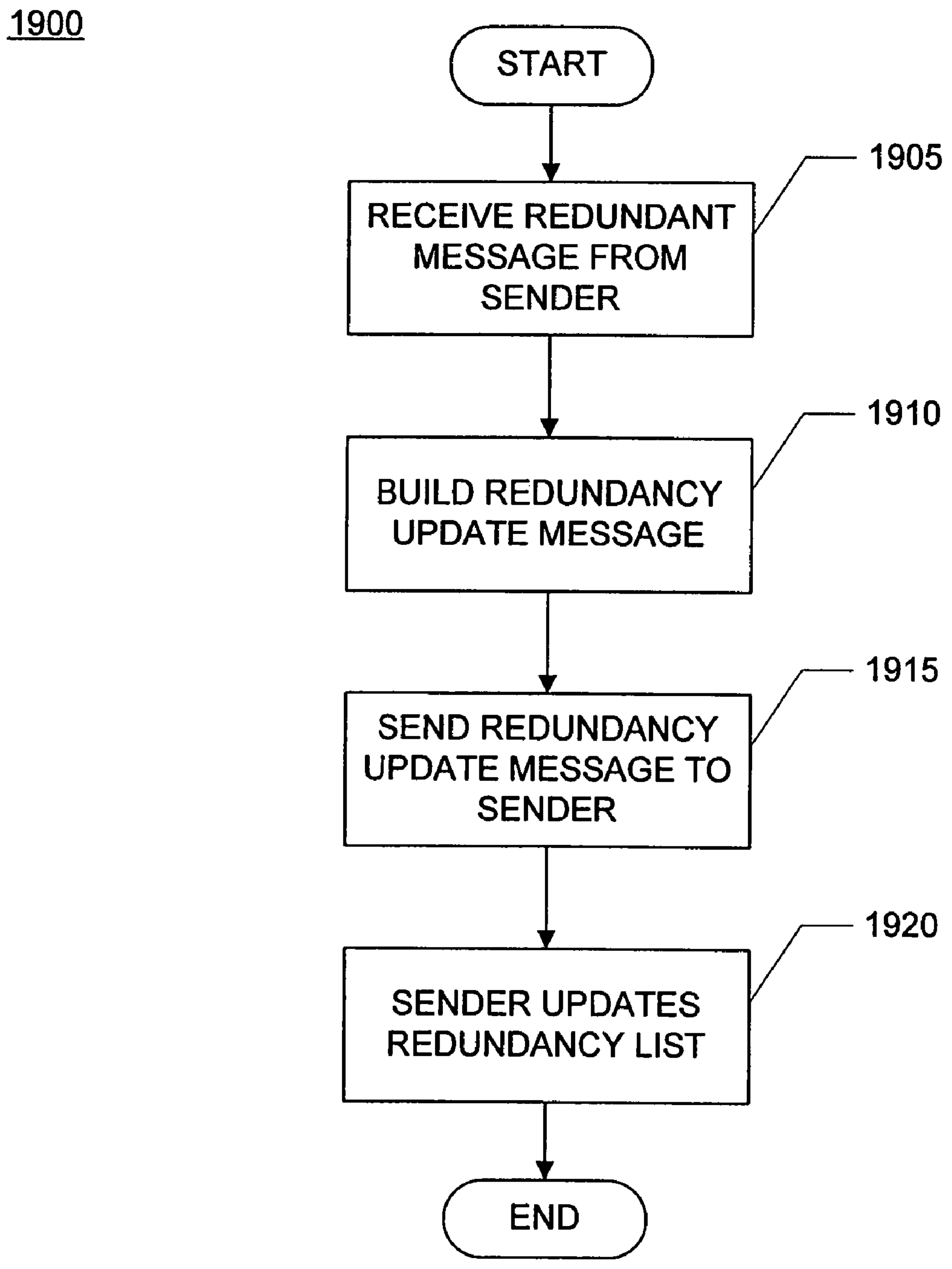

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

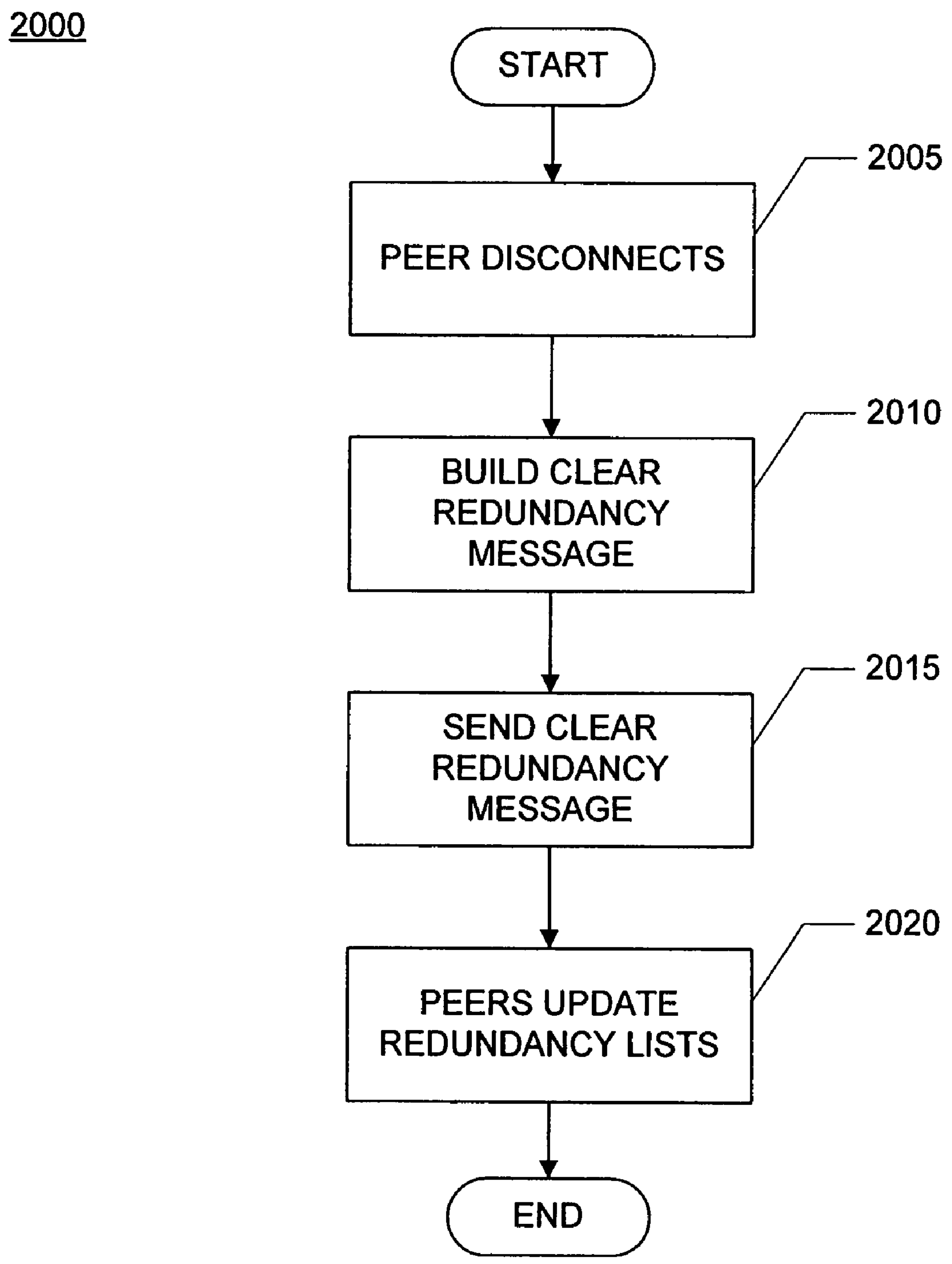

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

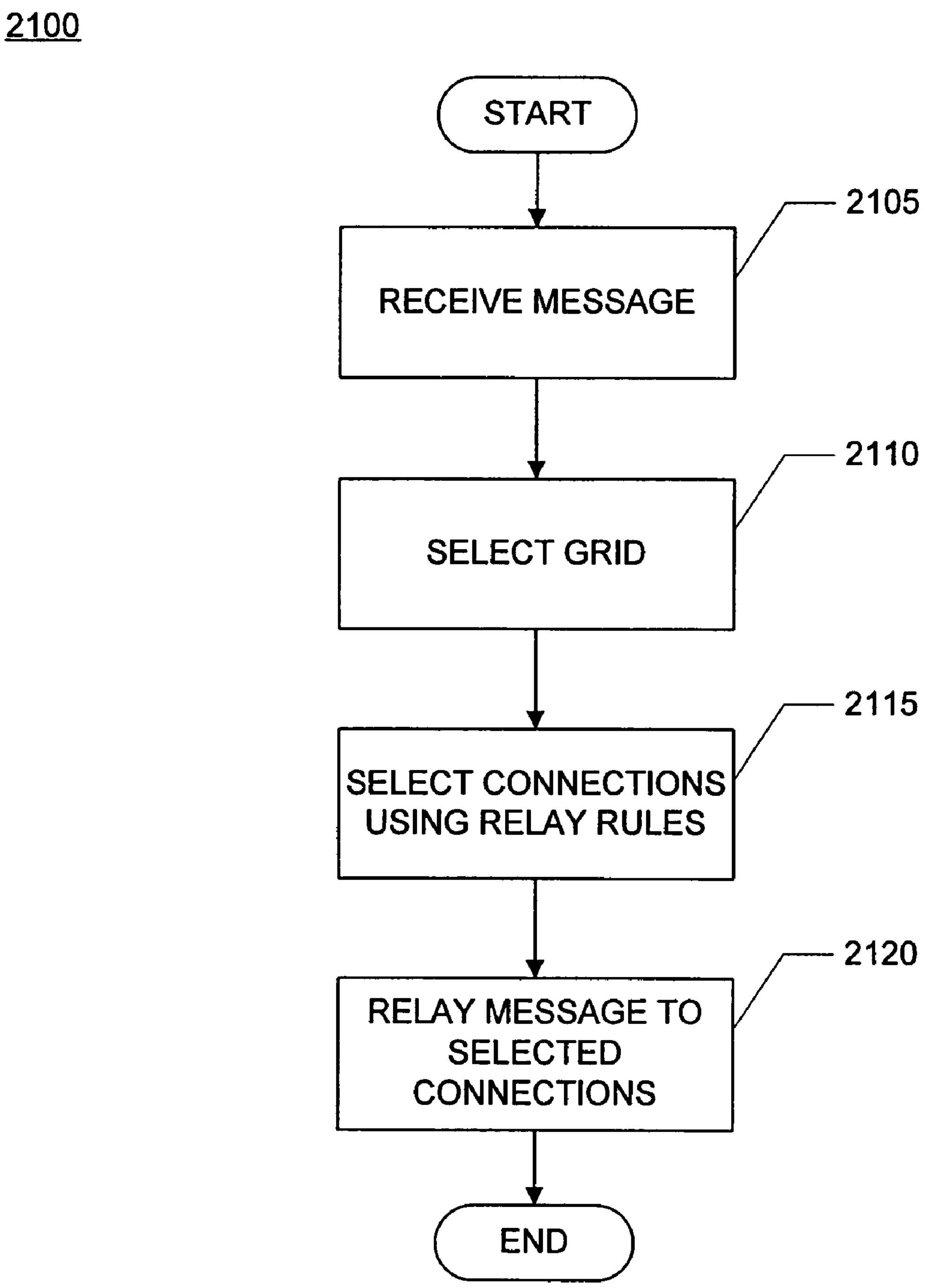

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

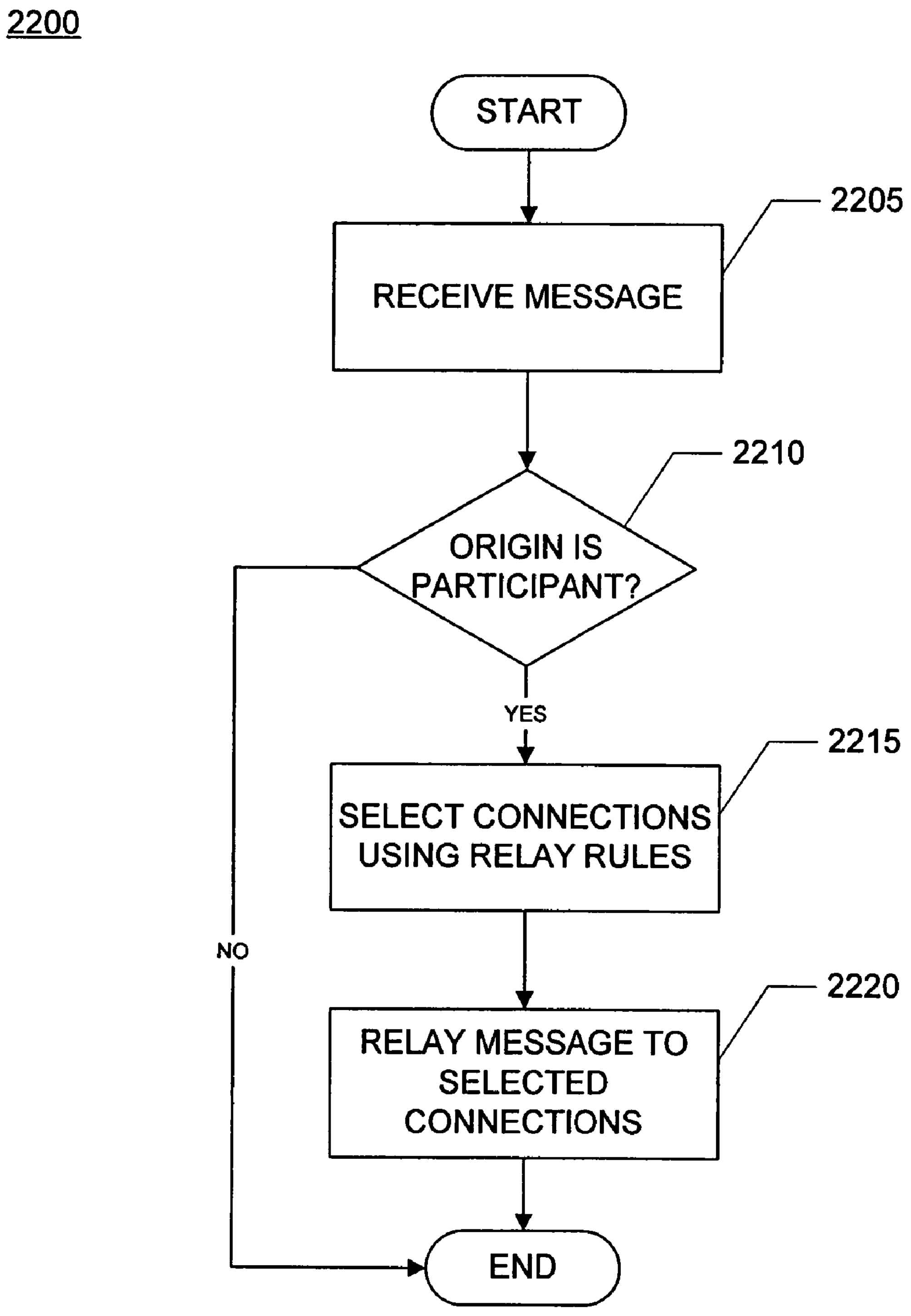

2200
START
2205
RECEIVE MESSAGE
2210
ORIGIN IS PARTICIPANT?
YES
NO
2215
SELECT CONNECTIONS USING RELAY RULES
2220
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 23

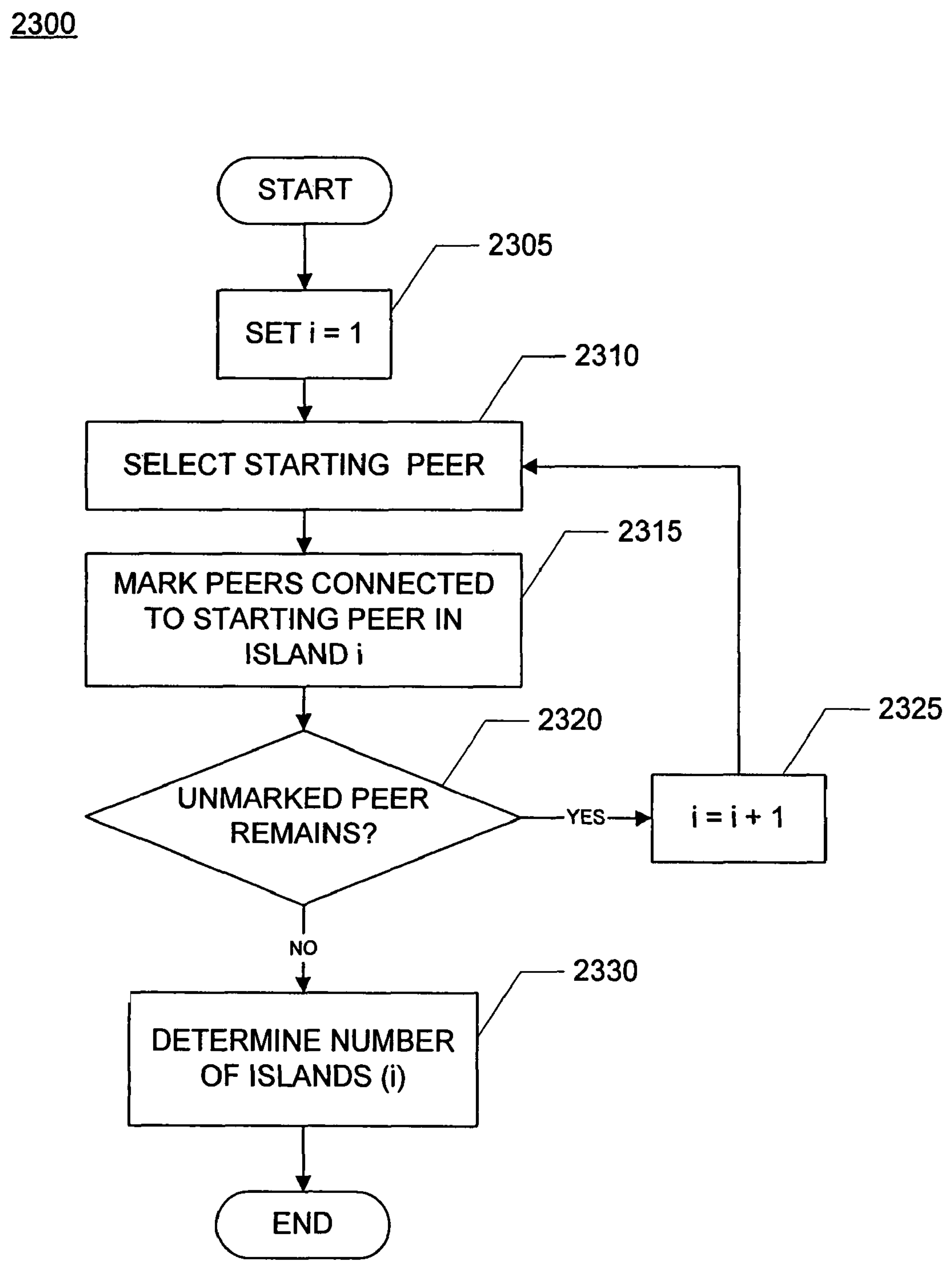

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

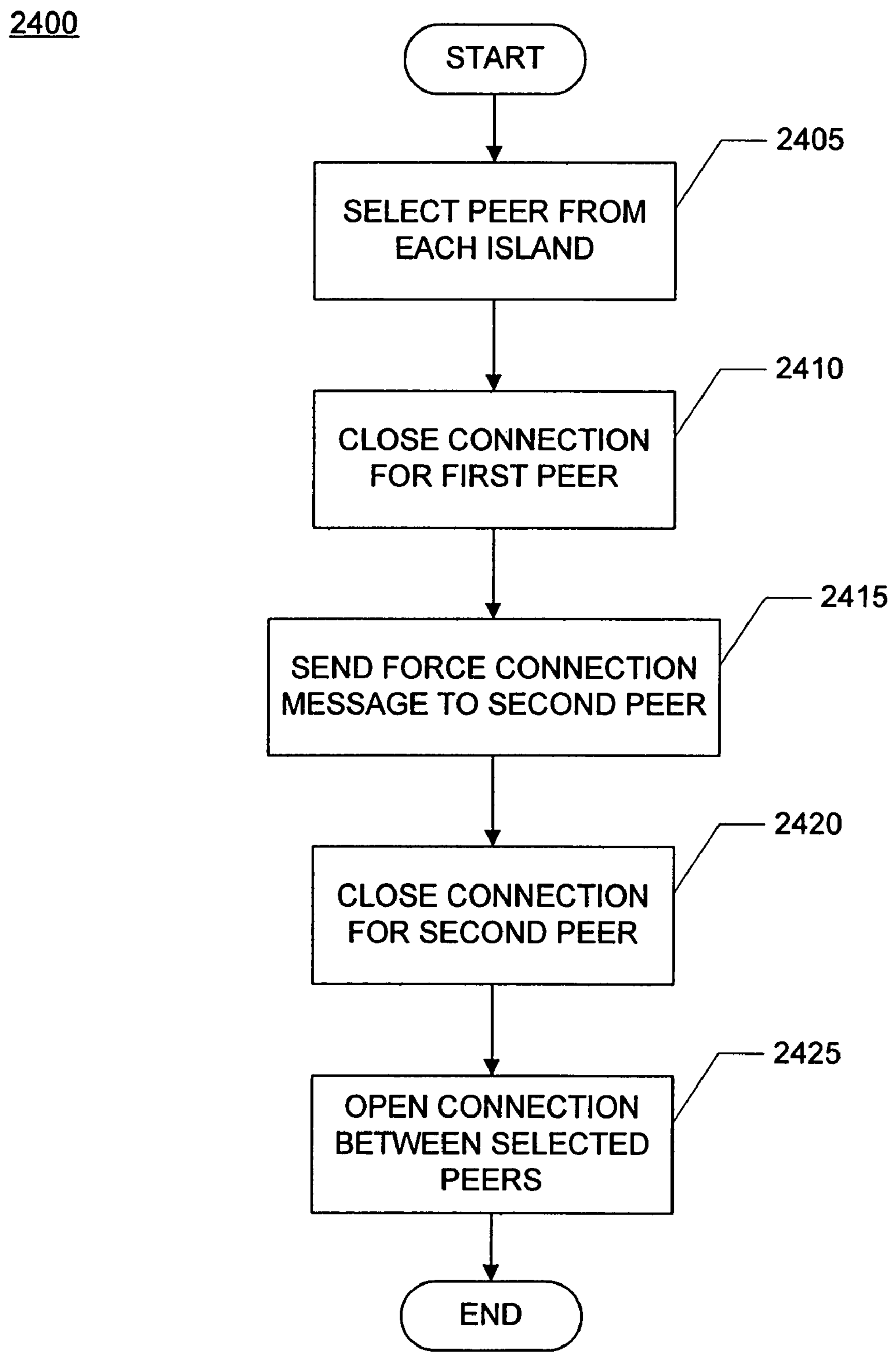

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

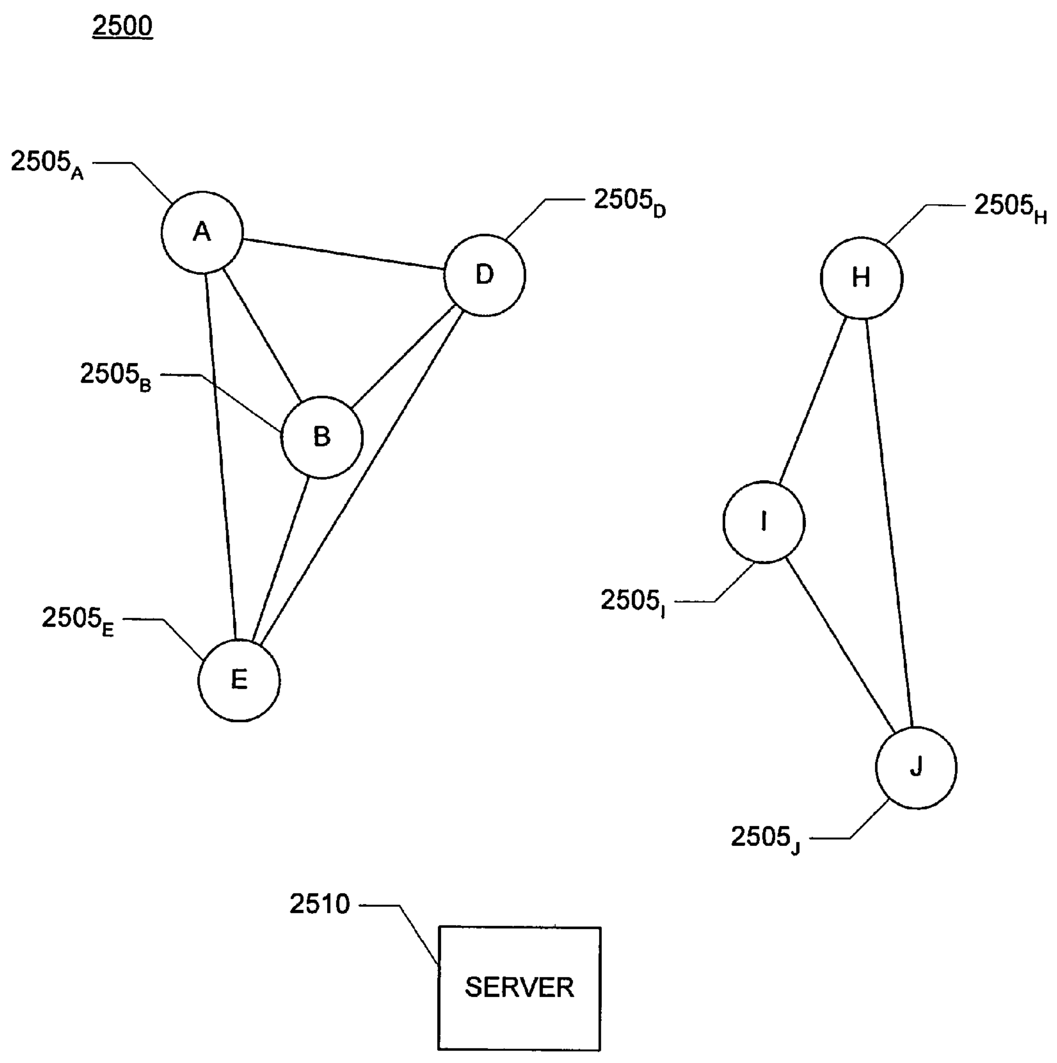

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 26

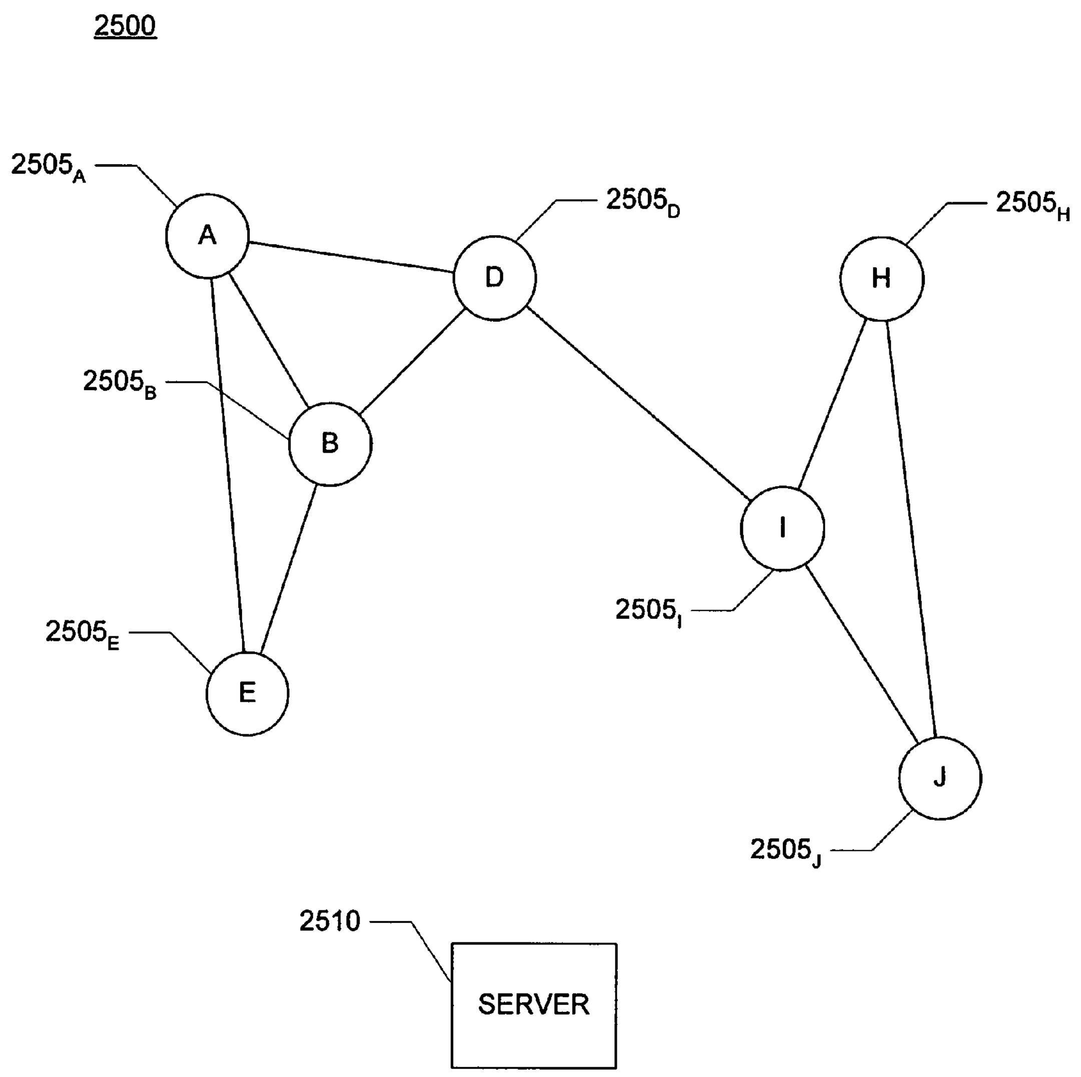

2500
2505_A
A
2505_D
D
2505_H
H
2505_B
B
I
2505_I
2505_E
E
J
2505_J
2510
SERVER

FIG. 27

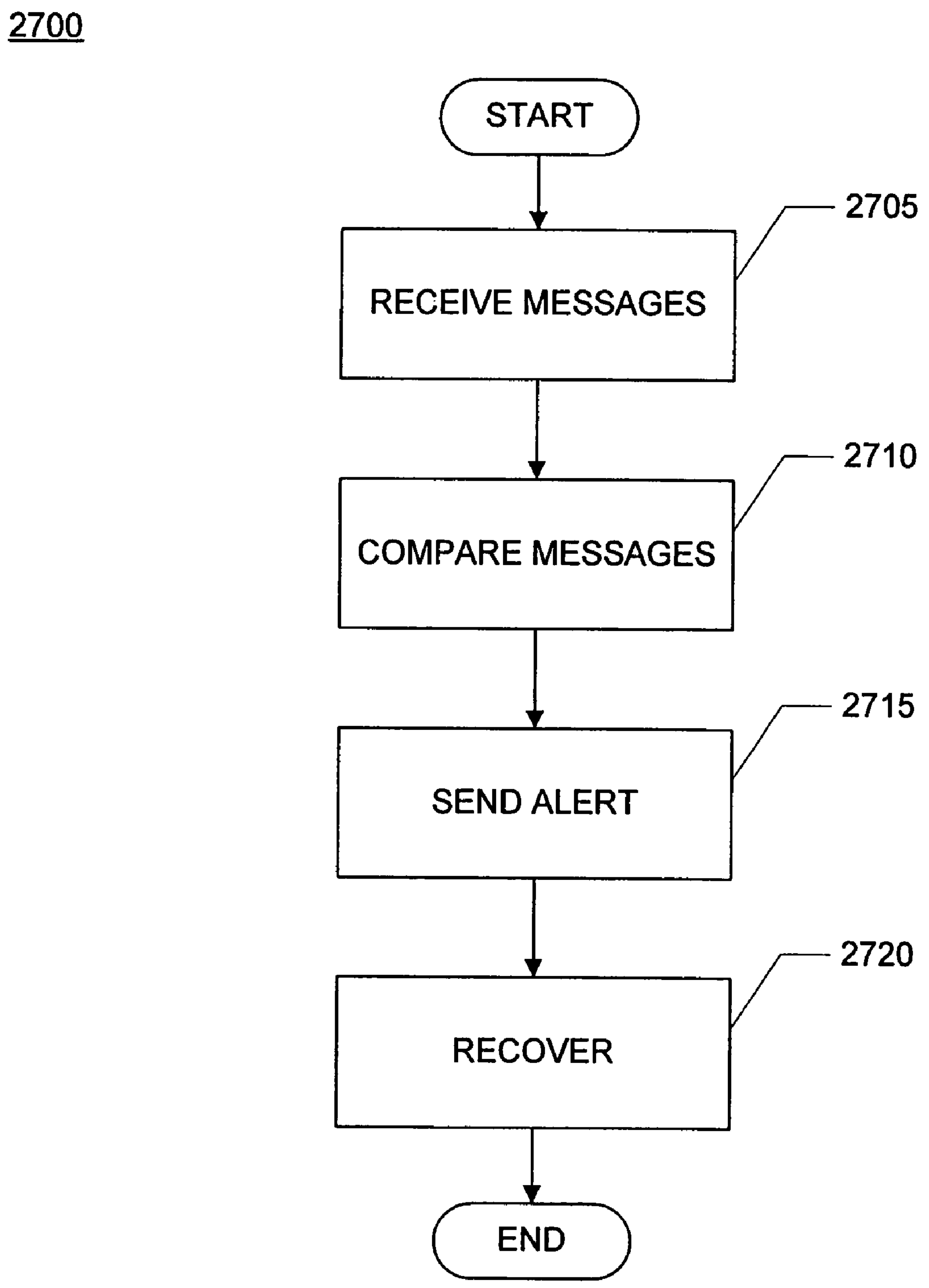

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

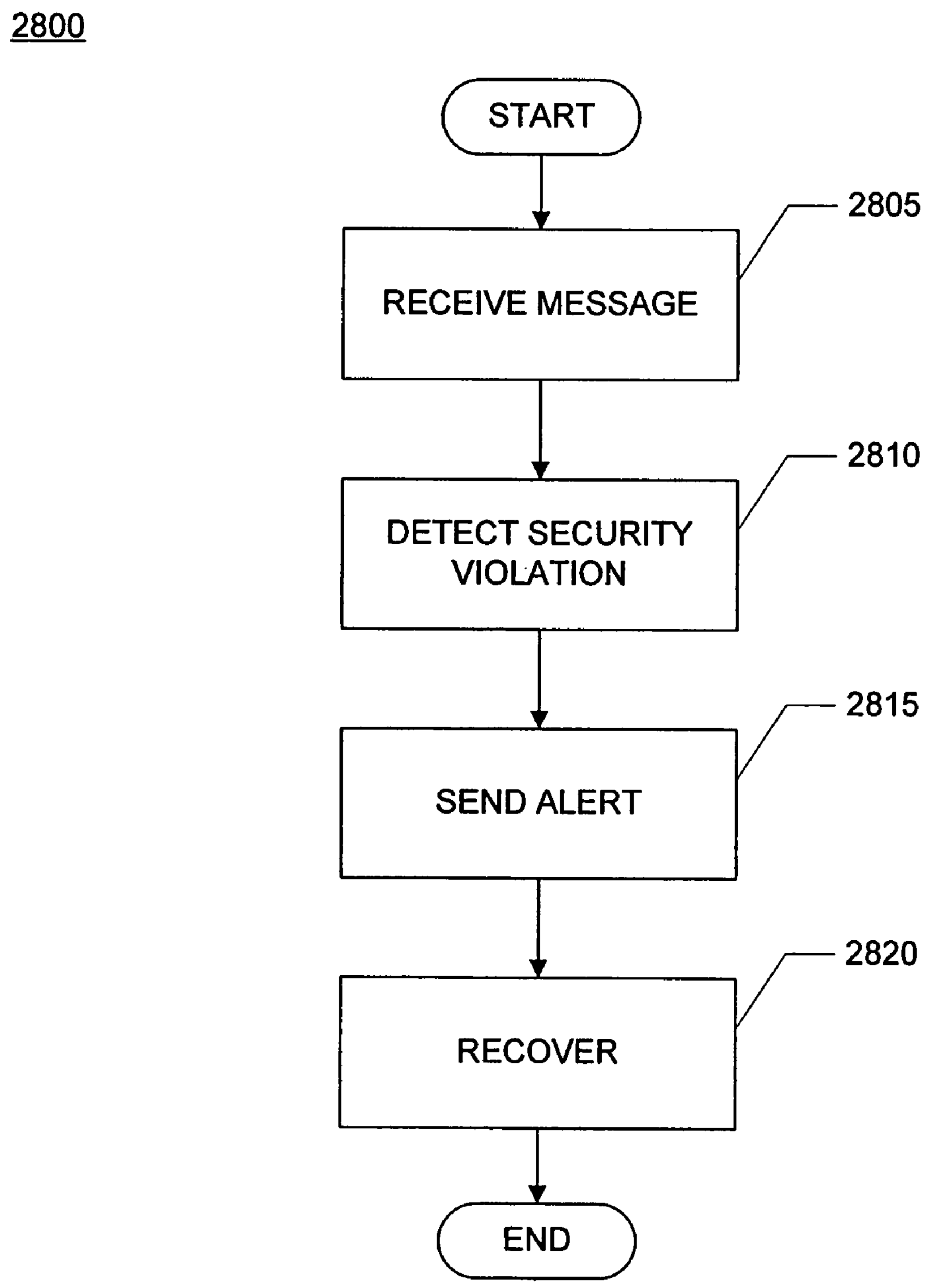

2800
START
2805
RECEIVE MESSAGE
2810
DETECT SECURITY VIOLATION
2815
SEND ALERT
2820
RECOVER
END

FIG. 29

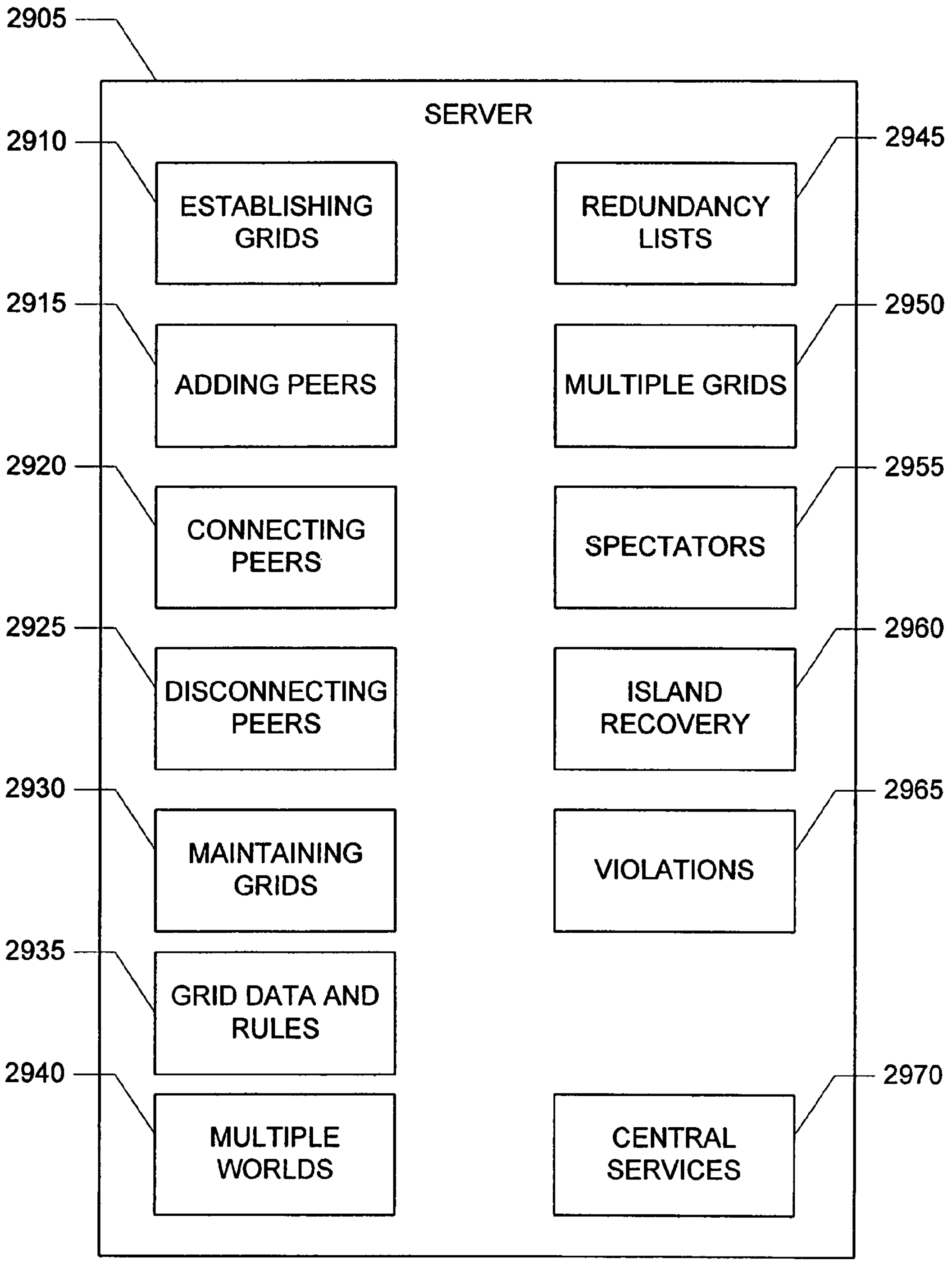

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

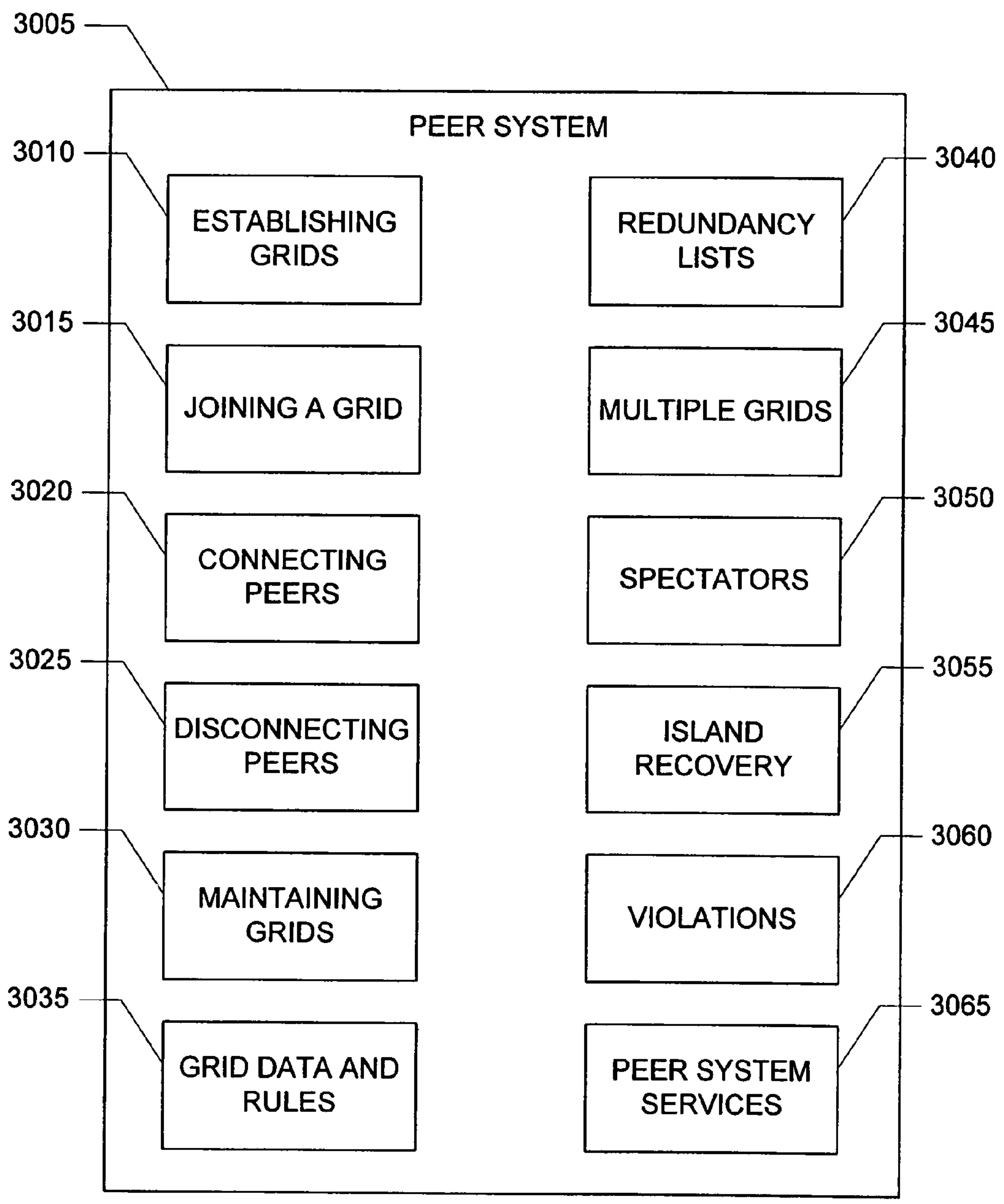

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

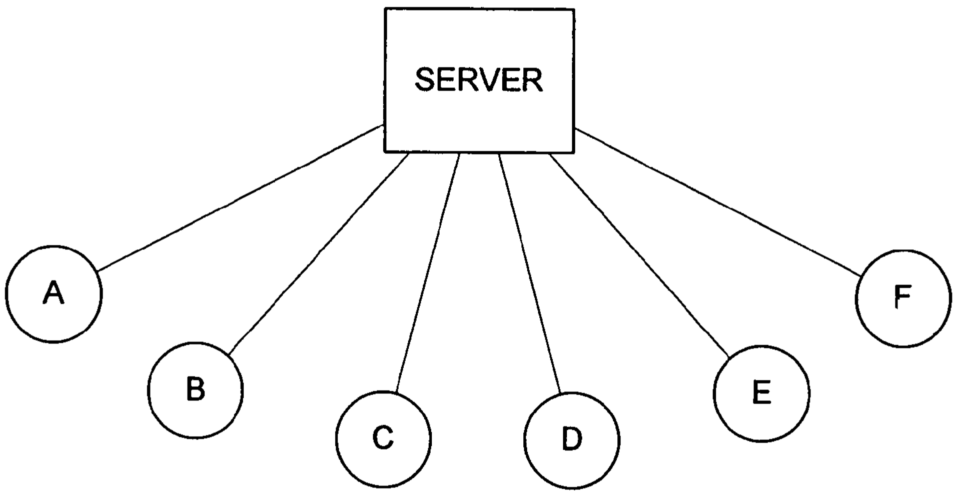
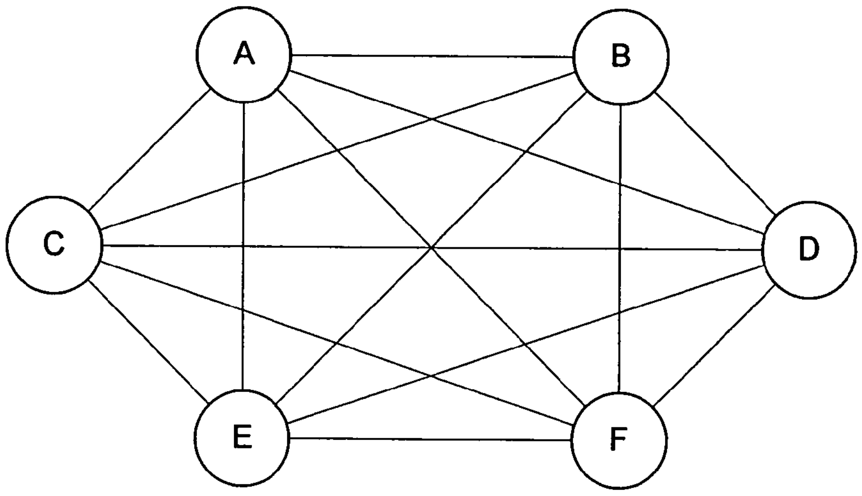


FIG. 31B



METHOD OF MAINTAINING A PEER-TO-PEER RELAY NETWORK

This is a division of application Ser. No. 10/700,798, filed Nov. 3, 2003, now U.S. Pat. No. 7,627,678 which claims benefit to U.S. Provisional Application No. 60/513,098; filed Oct. 20, 2003, the entirety of which is incorporated herein by reference.

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N−1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N−2, and each peer system is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules.

In another implementation, a server for a peer-to-peer relay network includes: means for establishing a peer-to-peer relay network; means for adding a peer system to a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for tracking connections in a peer-to-peer relay network.

In another implementation, a peer system for a peer-to-peer relay network includes: means for relaying data to any other peer systems connected to said peer system in a peer-to-peer relay network; means for establishing a peer-to-peer relay

network; means for joining a peer-to-peer relay network; means for connecting to another peer system in a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for disconnecting from another peer system connected to said peer system in a peer-to-peer relay network.

In another implementation, a method of relaying data in a peer-to-peer relay network includes: receiving data at a relaying peer system from a sending peer system connected to said relaying peer system in a peer-to-peer relay network; applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and relaying said data to any peer systems selected by applying said set of one or more relay rules.

In another implementation, a method of adding a peer system to a peer-to-peer relay network includes: opening a connection between a server and a joining peer system; providing grid information to said joining peer system indicating one or more established peer-to-peer relay networks; receiving a grid selection from said joining peer system indicating a selected peer-to-peer relay network, wherein said selected peer-to-peer relay network has one or more member peer systems; providing network addresses of each of said one or more member peer systems to said joining peer system; and receiving a connection update from said joining peer system indicating to which member peer systems said joining peer system is connected; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to a connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of joining a peer-to-peer relay network includes: sending a join message from a joining peer system to each of one or more member peer systems in a peer-to-peer relay network; receiving a join response from at least one of said one or more member peer systems, wherein each join response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting one or more member peer systems up to a connection limit according to a set of one or more connection rules; opening a connection with each selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of establishing a peer-to-peer relay network includes: opening a connection between said server and an establishing peer system, wherein the establishing peer system is one of said member peer systems; sending a request to create said peer-to-peer relay network from said establishing peer system to said server; receiving a creation confirmation at said establishing peer system from said server; wherein said establishing peer system stores a connection limit defining a number of other peer systems up to which said establishing peer system is permitted to connect, and said establishing peer system stores a set of one or more relay rules for relaying data to other peer systems connected to said establishing peer system.

In another implementation, a method of connecting peer systems in a peer-to-peer relay network includes: sending a connection available message from a disconnected peer system to one or more member peer systems in a peer-to-peer relay network when said disconnected peer system has a number of open connections to member systems that is less than a connection limit; receiving a connection available response from at least one of said one or more member peer systems, wherein each connection available response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting a member peer system according to a set of one or more connection rules; opening a connection with said selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of maintaining a peer-to-peer relay network includes: sending a maintenance message from a peer system to each of one or more connected peer systems connected to said peer system in a peer-to-peer relay network; evaluating any responses received from said one or more connected peer systems; and closing the connection between said peer system and a connected peer system if the response from that connected peer system is not acceptable; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit and each peer system stores a set of one or more relay rules for relaying data to the other peer systems connected to that peer system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems $\mathbf{105}_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent application Ser. Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed Jul. 31, 2002, and 10/211,129 ("Multi-User Application Programming Interface"), filed Jul. 31, 2002, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system $\mathbf{105}_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and

compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an

indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members; not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the

member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new

peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet).

In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins,

peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B.

Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method of maintaining a peer-to-peer relay network, comprising:

sending a maintenance message from a peer computer system to each of one or more connected peer computer systems connected to said peer computer system in a peer-to-peer relay network;

evaluating any responses received from said one or more connected peer computer systems;

determining the connection between said peer computer system and a connected peer computer system is a failed connection when the response from that connected peer computer system is not acceptable;

sending, in response to determining a failed connection, a close connection request message to the connected peer computer system;

closing the connection between the peer computer system and a connected peer computer system when a confirmation is received from the connected peer computer system, and

in response to closing the connection, sending a connection available message to other peers in the peer-to-peer relay network,

wherein when the peer-to-peer network is connected to a server, sending the connection available message to the server,

wherein each peer computer system is connected to a number of other peer computer systems that is less than or equal to a connection limit and each peer computer system stores a set of one or more relay rules for relaying data to the other peer computer systems connected to that peer computer system.

2. The method of claim **1**, wherein:

said maintenance message is a ping message.

3. The method of claim **1**, wherein:

a response from a connected peer computer system is not acceptable if the response is not received by said peer computer system within a time limit.

4. The method of claim **1**, wherein:

a response is considered not acceptable if the response is not received.

5. The method of claim **1**, wherein:

a response from a connected peer computer system is considered not acceptable if said peer computer system has not received a response from that connected peer computer system within a time limit multiple times.

6. The method of claim **1**, further comprising:

sending a connection status request to a server for a connection between said peer computer system and a connected peer computer system if the response from that connected peer computer system is not acceptable.

7. The method of claim **1**, further comprising:

sending an update to a server indicating a connection has been closed for each connection closed by said peer computer system.

* * * * *

EXHIBIT 92

US008010633B2

(12) **United States Patent**
Van Datta et al.

(10) **Patent No.: US 8,010,633 B2**
(45) **Date of Patent: Aug. 30, 2011**

(54) # MULTIPLE PEER-TO-PEER RELAY NETWORKS

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America LLC**, Foster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1785 days.

(21) Appl. No.: **10/701,302**

(22) Filed: **Nov. 3, 2003**

(65) **Prior Publication Data**

US 2005/0086329 A1 Apr. 21, 2005

Related U.S. Application Data

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/177 (2006.01)

(52) **U.S. Cl.** **709/220**

(58) **Field of Classification Search** 709/220
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,787,051 A 11/1988 Olson

(Continued)

FOREIGN PATENT DOCUMENTS

EP 0 913 965 5/1999

(Continued)

OTHER PUBLICATIONS

Fox et al; Federated Grids and their Security; Available on the Wayback Machine as of Apr. 20, 2003; Retrieved from http://web.archive.org/web/20051013044806/grids.ucs.indiana.edu/ptliupages/publications/FedGrid_Short. pdf.*

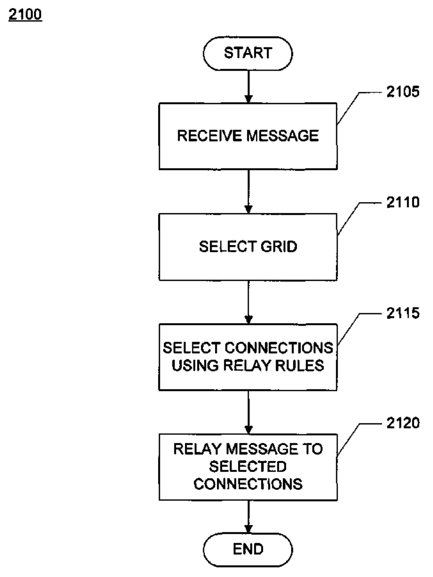
(Continued)

Primary Examiner — Carl Colin
Assistant Examiner — Chau D Le
(74) *Attorney, Agent, or Firm* — Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a network environment supporting multiple peer-to-peer relay networks includes: a first peer-to-peer relay network including N1 peer systems; and a second peer-to-peer relay network including N2 peer systems; wherein each peer system in said first peer-to-peer relay network is connected to a number of other peer systems in said first peer-to-peer relay network that is less than or equal to a first connection limit, said first connection limit is greater than or equal to 2, said first connection limit is less than or equal to N1-2, each peer system in said first peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a first set of one or more relay rules, each peer system in said second peer-to-peer relay network is connected to a number of other peer systems in said second peer-to-peer relay network that is less than or equal to a second connection limit, said second connection limit is greater than or equal to 2, said second connection limit is less than or equal to N2-2, each peer system in said second peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a second set of one or more relay rules, and at least one peer system in said first peer-to-peer relay network is also in said second peer-to-peer relay network.

20 Claims, 31 Drawing Sheets


2100
START
RECEIVE MESSAGE
2105
SELECT GRID
2110
SELECT CONNECTIONS USING RELAY RULES
2115
RELAY MESSAGE TO SELECTED CONNECTIONS
2120
END

U.S. PATENT DOCUMENTS

4,843,568	A	6/1989	Krueger	
5,128,671	A	7/1992	Thomas, Jr.	
5,528,265	A	6/1996	Harrison	
6,157,368	A	12/2000	Faeger	
6,375,572	B1	4/2002	Masuyama	
6,487,600	B1 *	11/2002	Lynch	709/229
7,240,093	B1 *	7/2007	Danieli et al.	709/205
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0085097	A1	7/2002	Colmenarez et al.	
2002/0119821	A1	8/2002	Sen et al.	
2002/0184310	A1 *	12/2002	Traversat et al.	709/204
2002/0184311	A1 *	12/2002	Traversat et al.	709/204
2003/0055892	A1	3/2003	Huitema et al.	
2003/0182421	A1 *	9/2003	Faybishenko et al.	709/224
2003/0191828	A1 *	10/2003	Ramanathan et al.	709/221
2003/0217135	A1 *	11/2003	Chatani et al.	709/223
2004/0212589	A1	10/2004	Hall et al.	

FOREIGN PATENT DOCUMENTS

EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

Song Jiang et al: "FloodTrial : an efficient file search technique in unstructured peer-to-peer systems" GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: "Impact of transmit range on throughput performance in mobile ad hoc networks" ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, Jun. 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y Ed—Association for Computing Machinery: "Simple and Fault-Tolerant Key Agreement by Dynamic Collaborative Groups" Proceedings of the 7th ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. Conf. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

Bolt, R.A., "Put-that-there": voice and gesture at the graphics interface, Computer Graphics, vol. 14, No. 3 (ACM SIGGRAPH Conference Proceedings) Jul. 1980, pp. 262-270.

DeWitt, Thomas and Edelstein, Phil, "Pantomation: A System for Position Tracking," Proceedings of the 2nd Symposium on Small Computers in the Arts, Oct. 1982, pp. 61-69.

* cited by examiner

FIG. 1

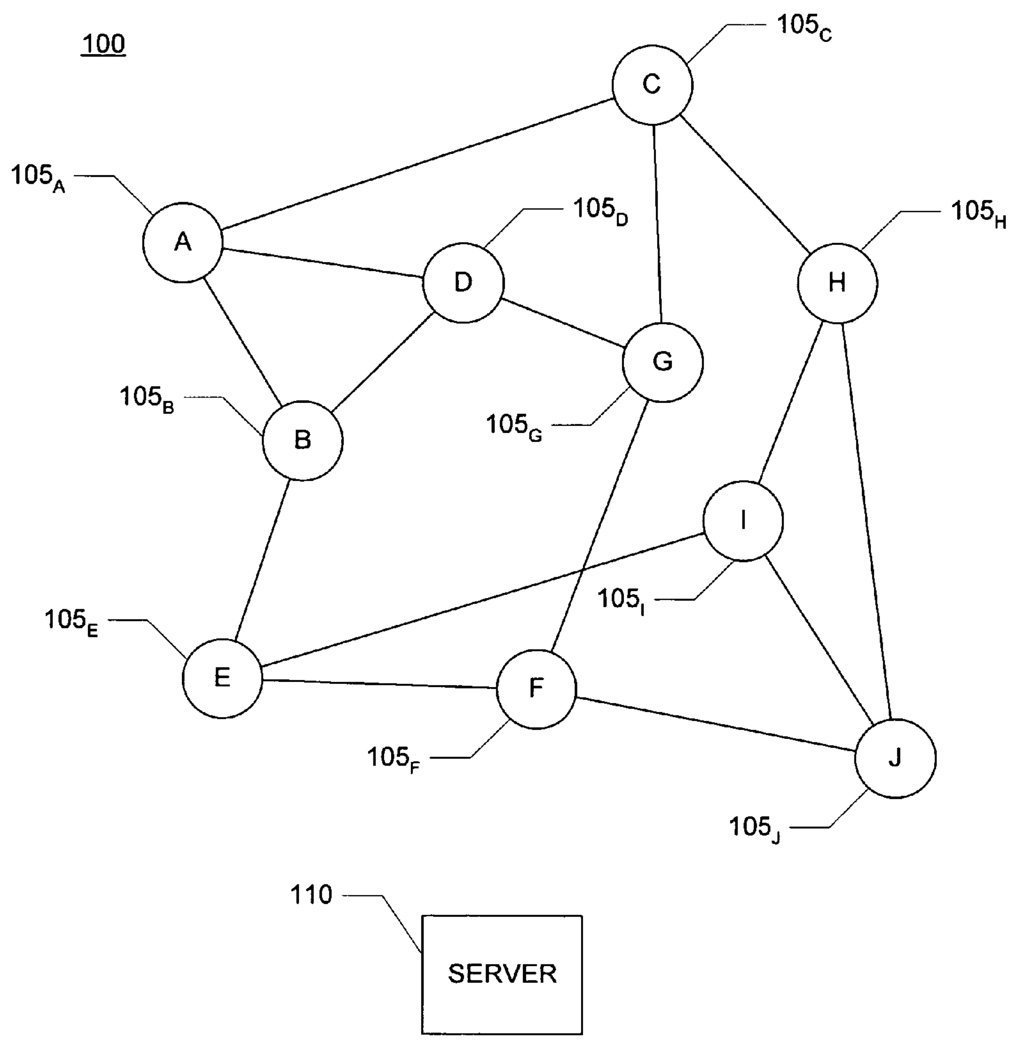

100
105C
C
105A
A
105D
D
105H
H
G
105B
B
105G
I
105I
105E
E
F
105F
J
105J
110
SERVER

FIG. 2

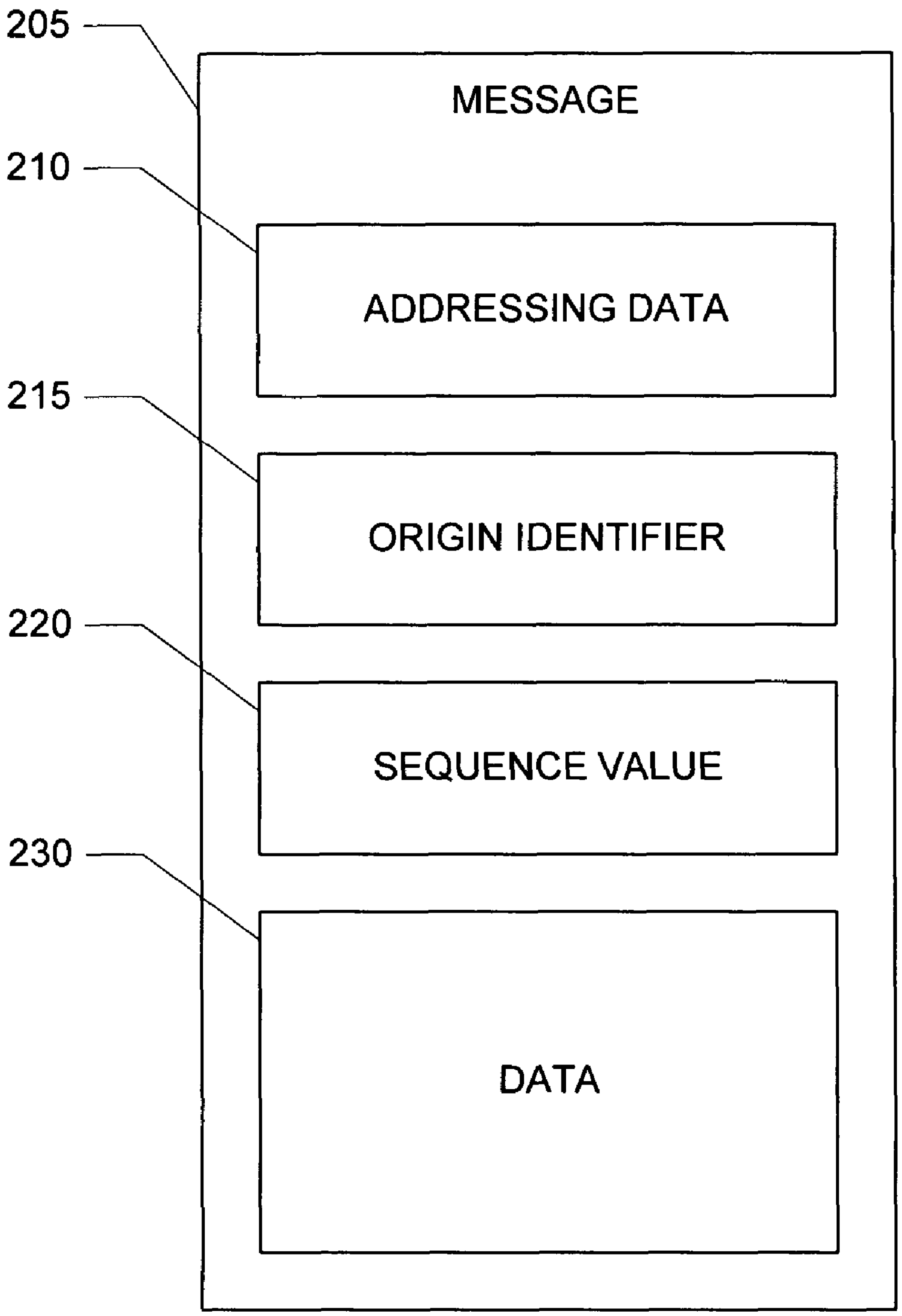

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

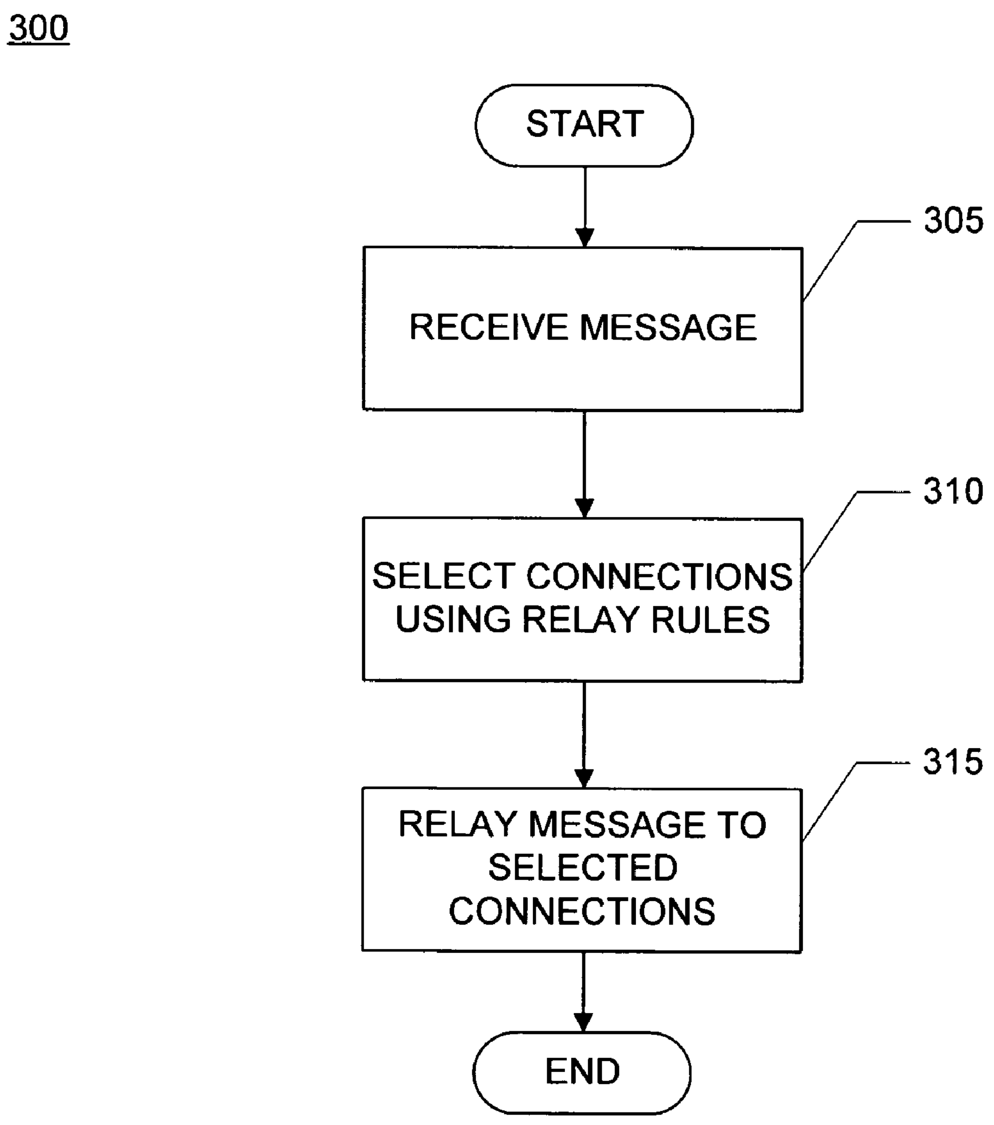

300
START
RECEIVE MESSAGE
305
SELECT CONNECTIONS USING RELAY RULES
310
RELAY MESSAGE TO SELECTED CONNECTIONS
315
END

FIG. 4

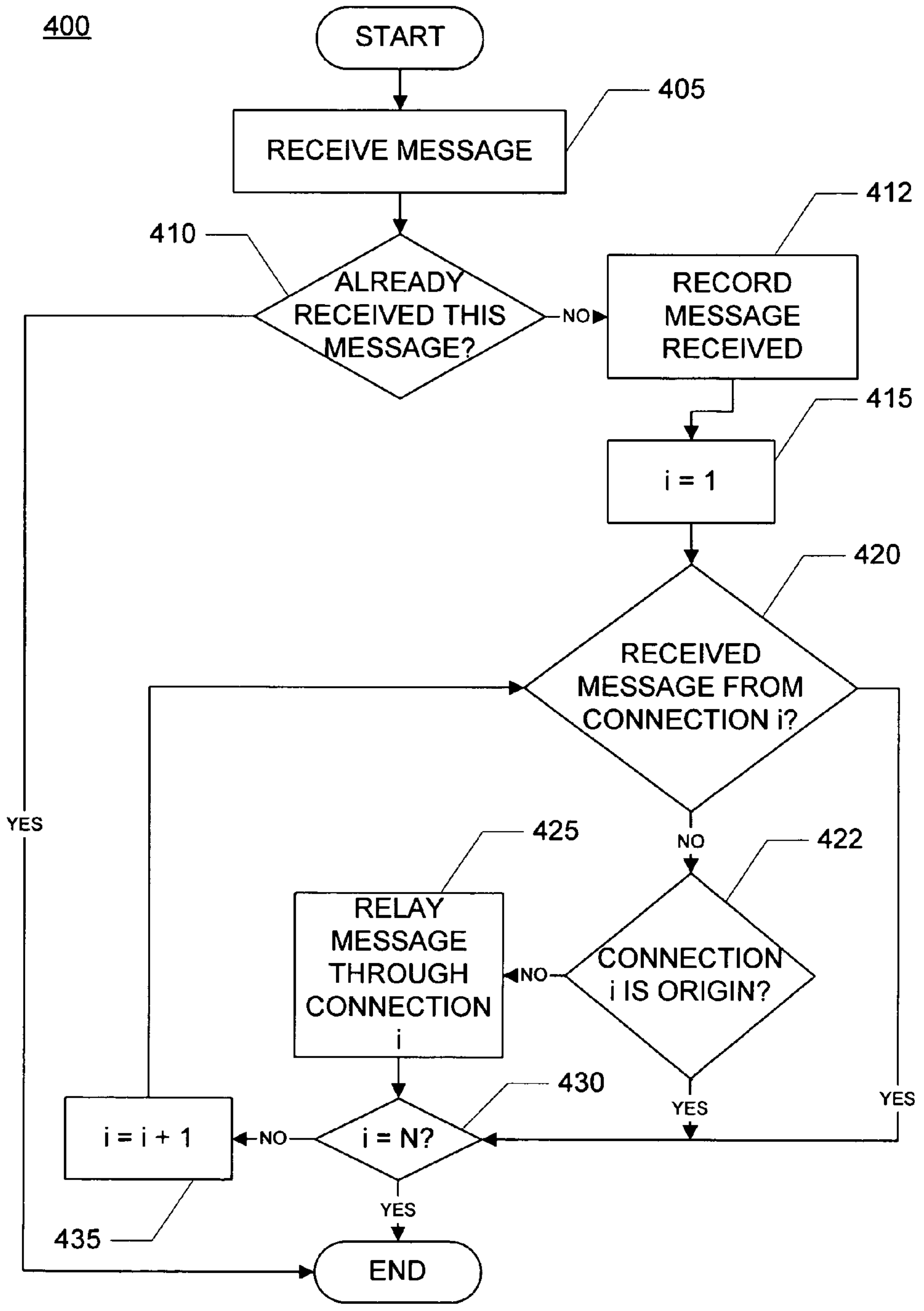

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
RECORD MESSAGE RECEIVED
412
YES
i = 1
415
RECEIVED MESSAGE FROM CONNECTION i?
420
NO
CONNECTION i IS ORIGIN?
422
NO
RELAY MESSAGE THROUGH CONNECTION i
425
YES
YES
i = N?
430
NO
i = i + 1
435
YES
END

FIG. 5

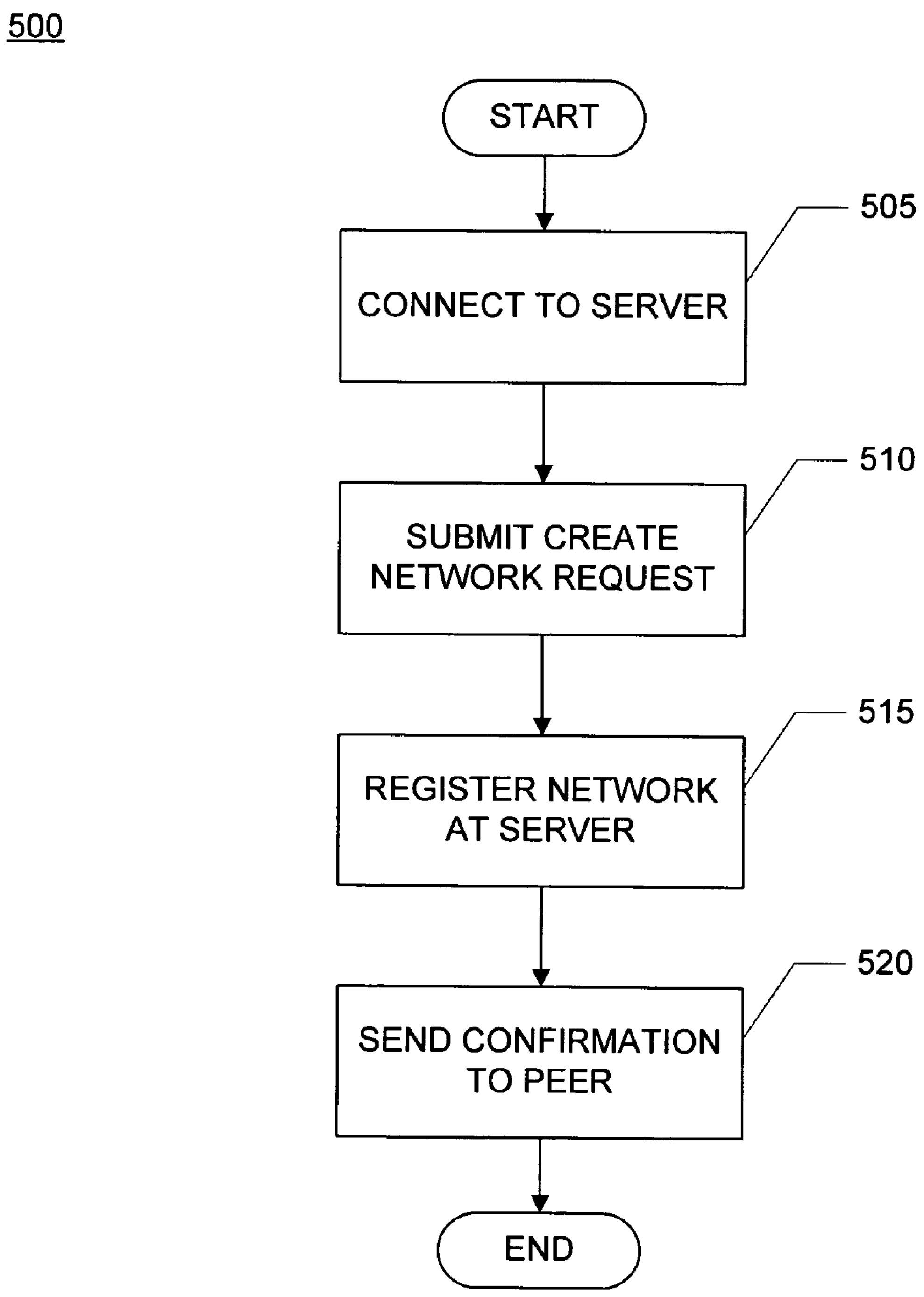

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

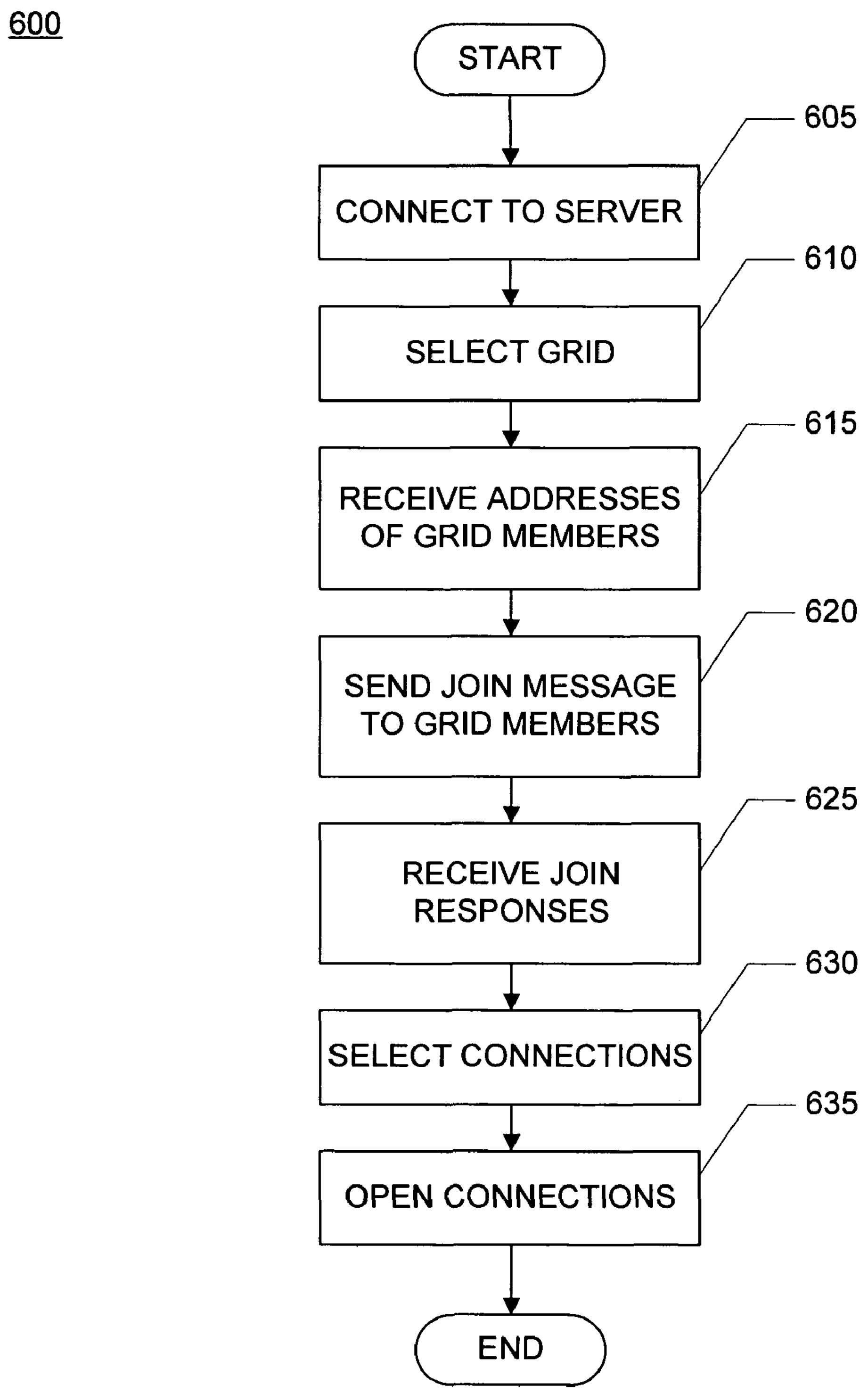

600
START
605
CONNECT TO SERVER
610
SELECT GRID
615
RECEIVE ADDRESSES OF GRID MEMBERS
620
SEND JOIN MESSAGE TO GRID MEMBERS
625
RECEIVE JOIN RESPONSES
630
SELECT CONNECTIONS
635
OPEN CONNECTIONS
END

FIG. 7

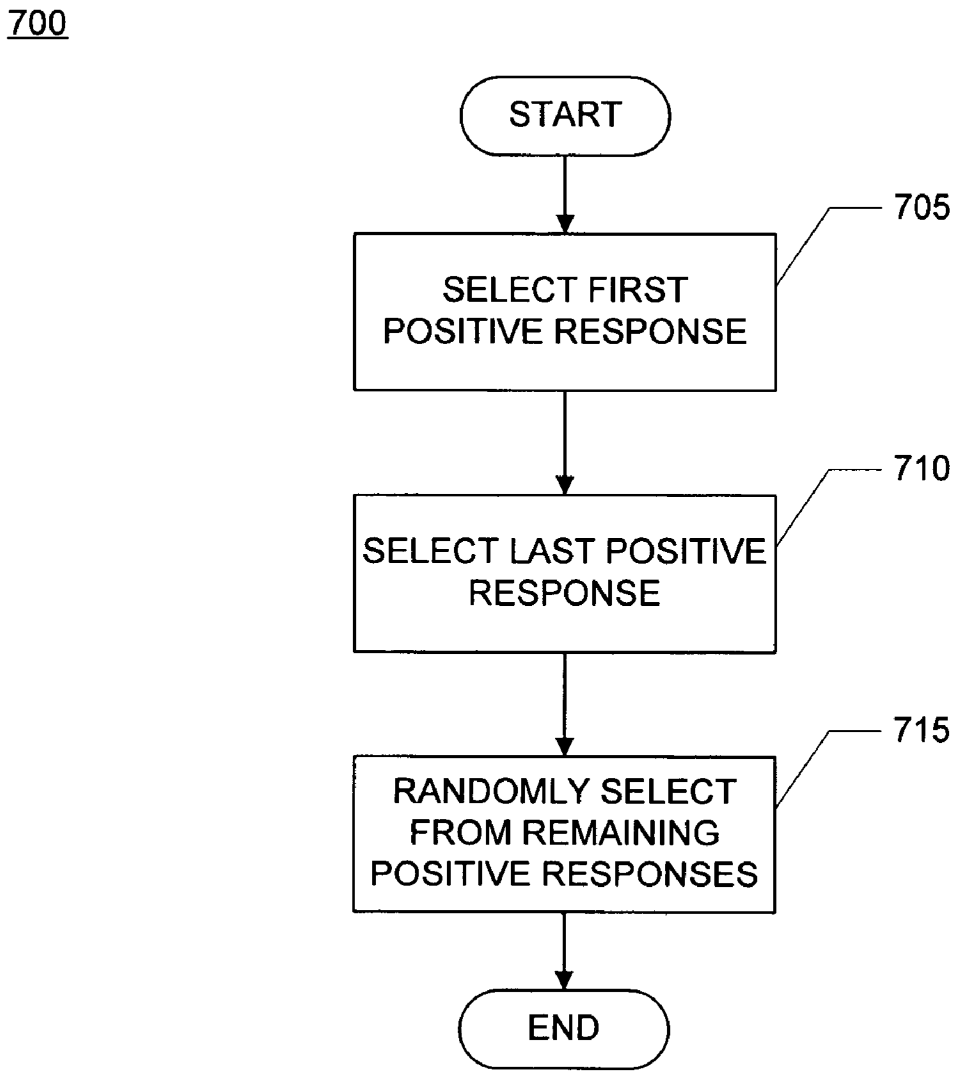

700
START
705
SELECT FIRST POSITIVE RESPONSE
710
SELECT LAST POSITIVE RESPONSE
715
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
END

FIG. 8

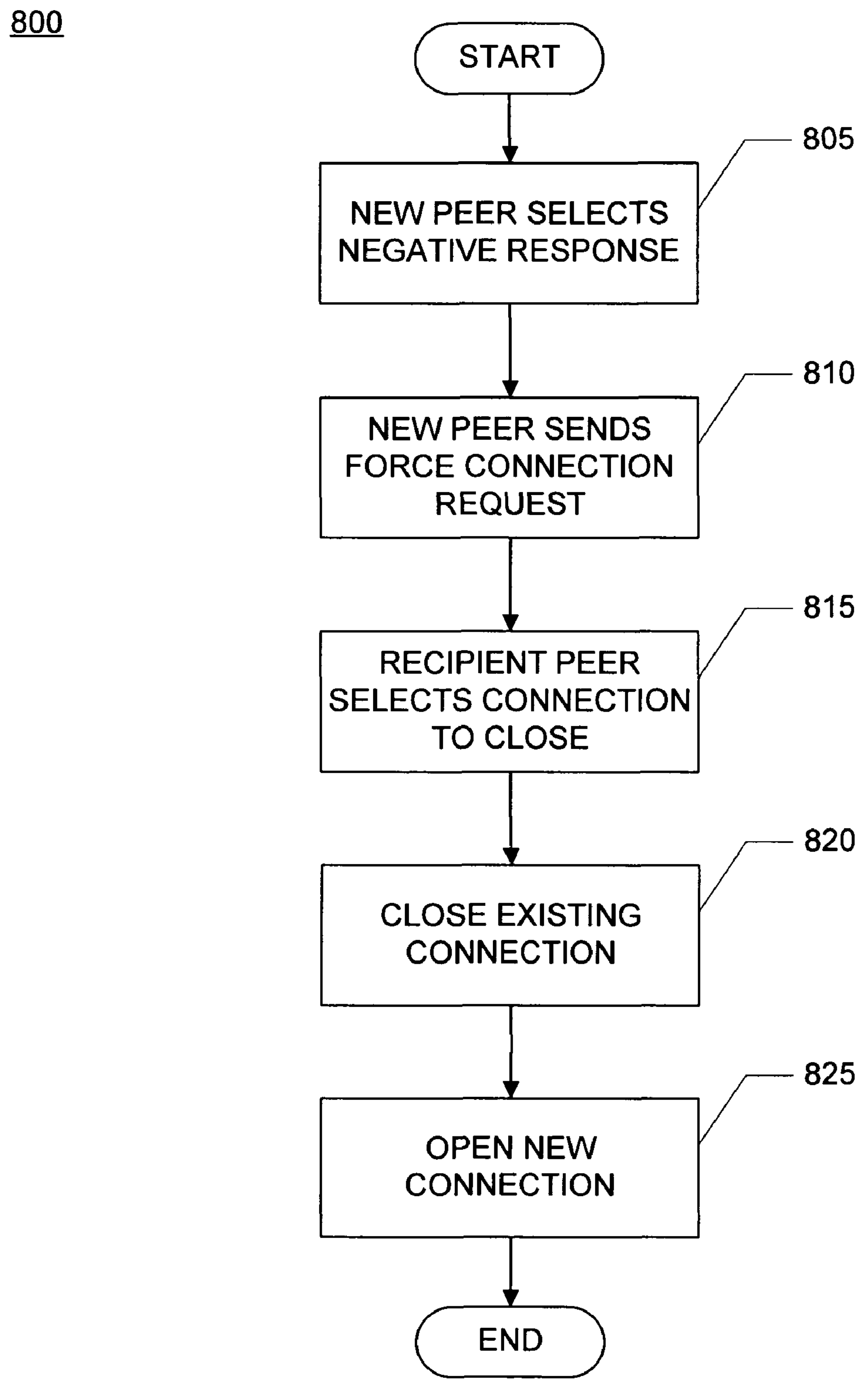

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

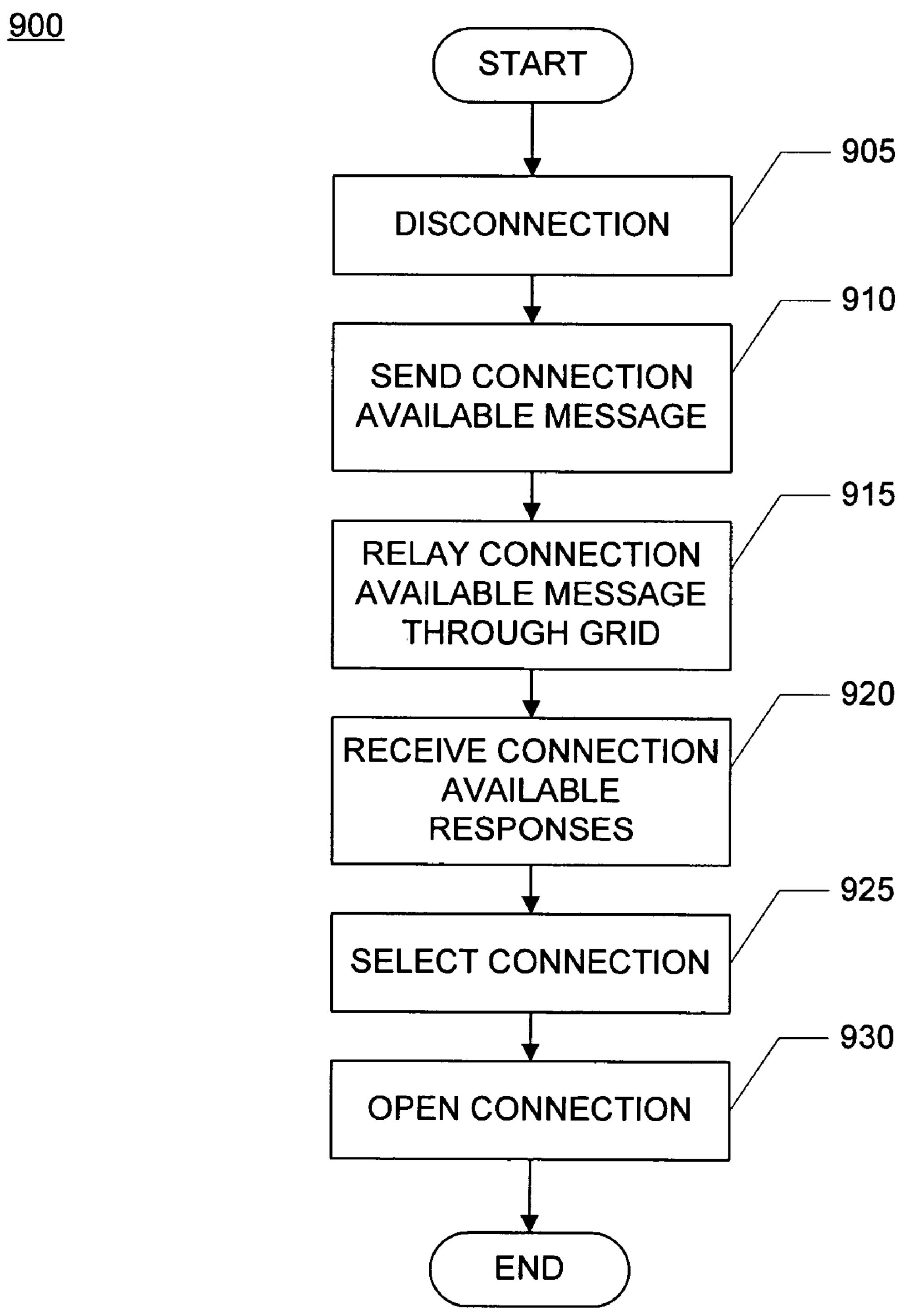

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

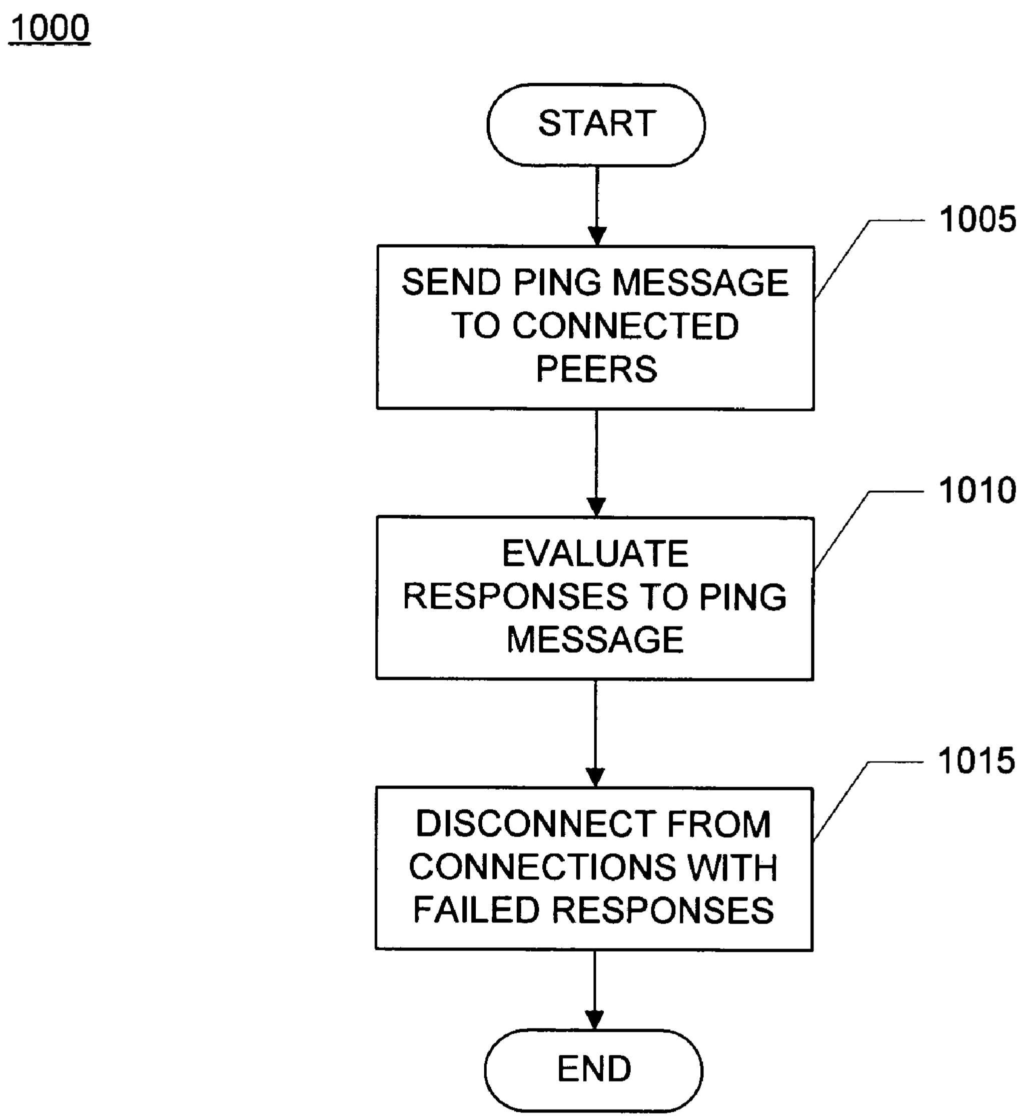

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

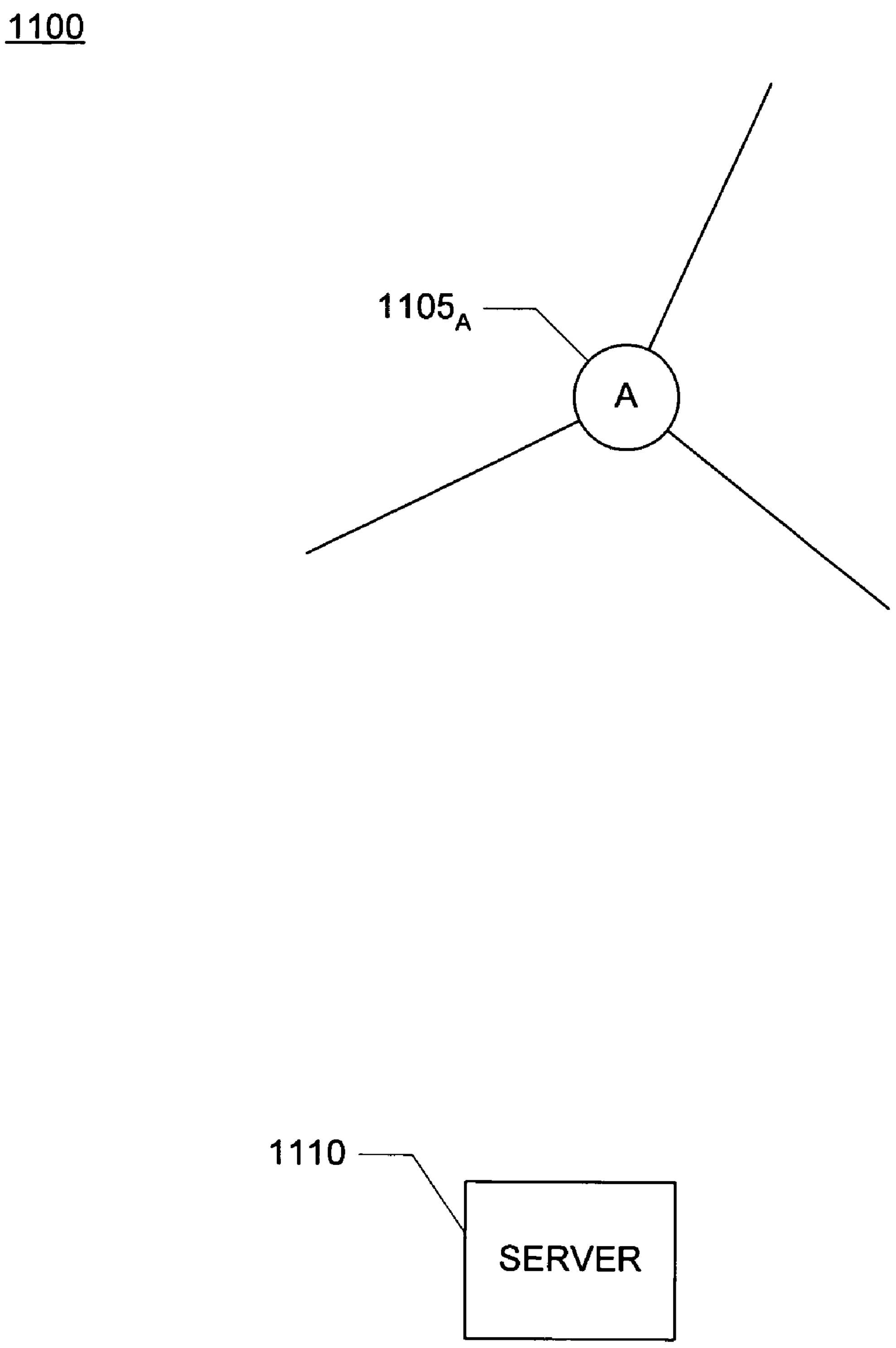


FIG. 12

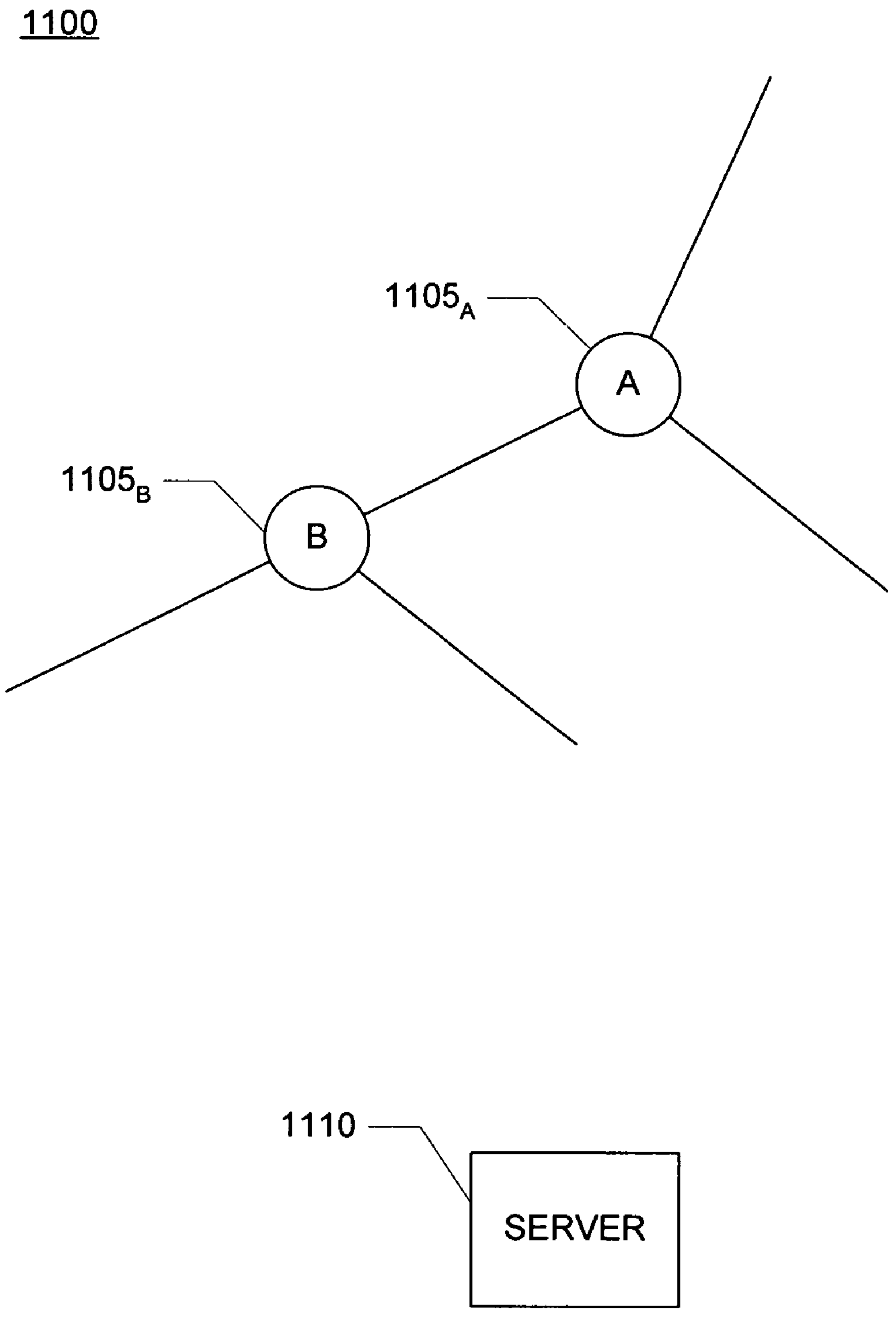


FIG. 13

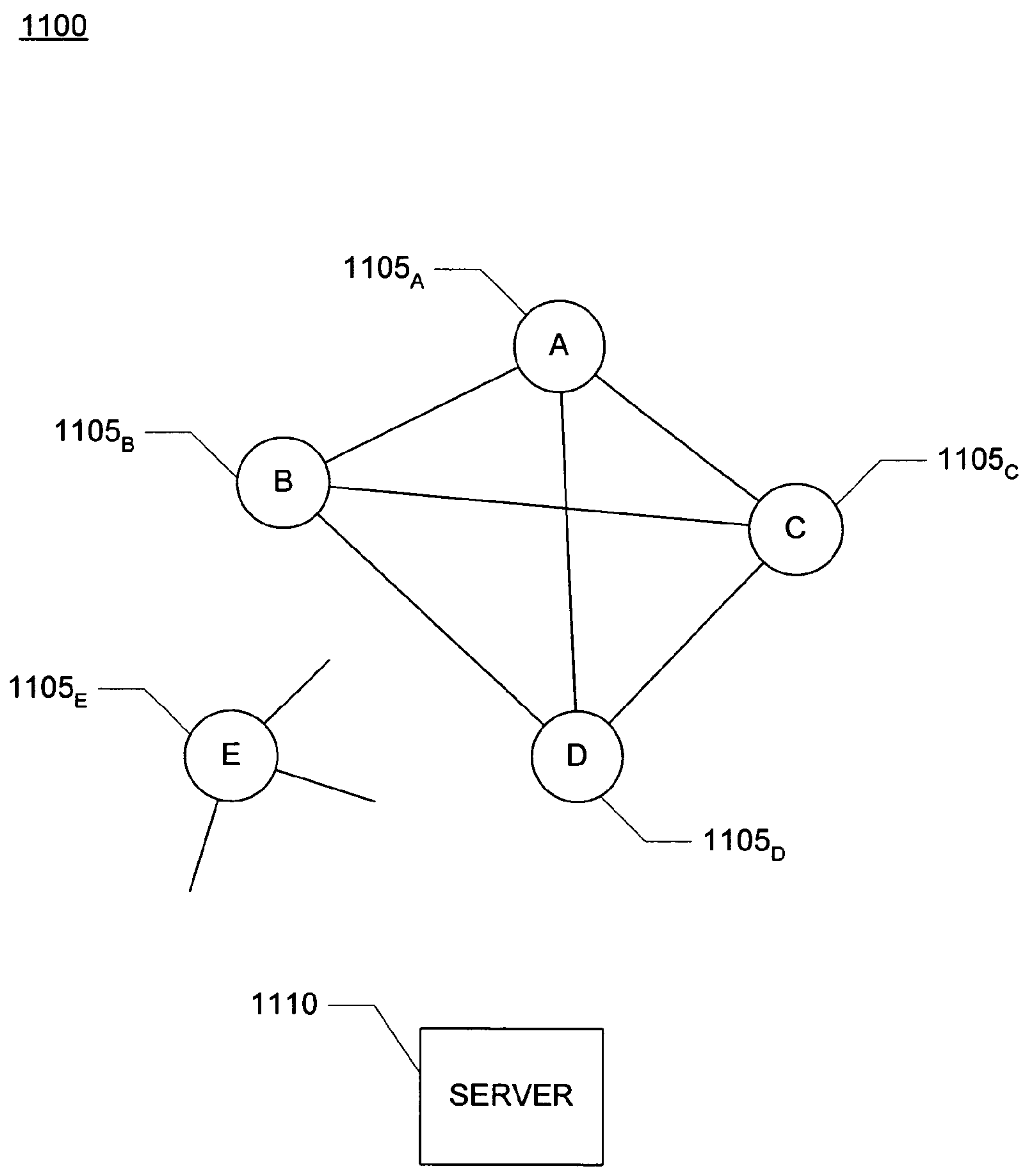


FIG. 14

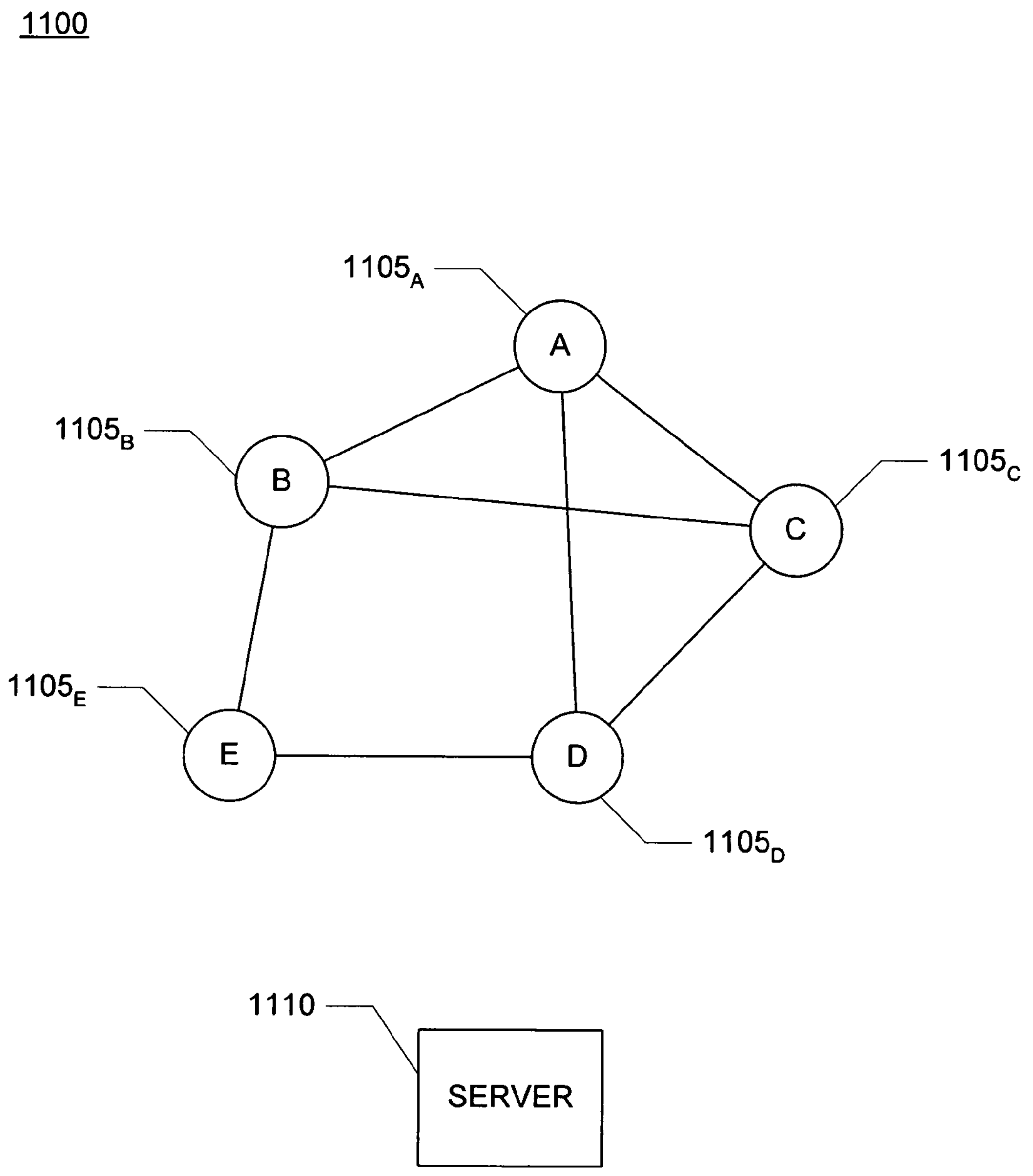


FIG. 15

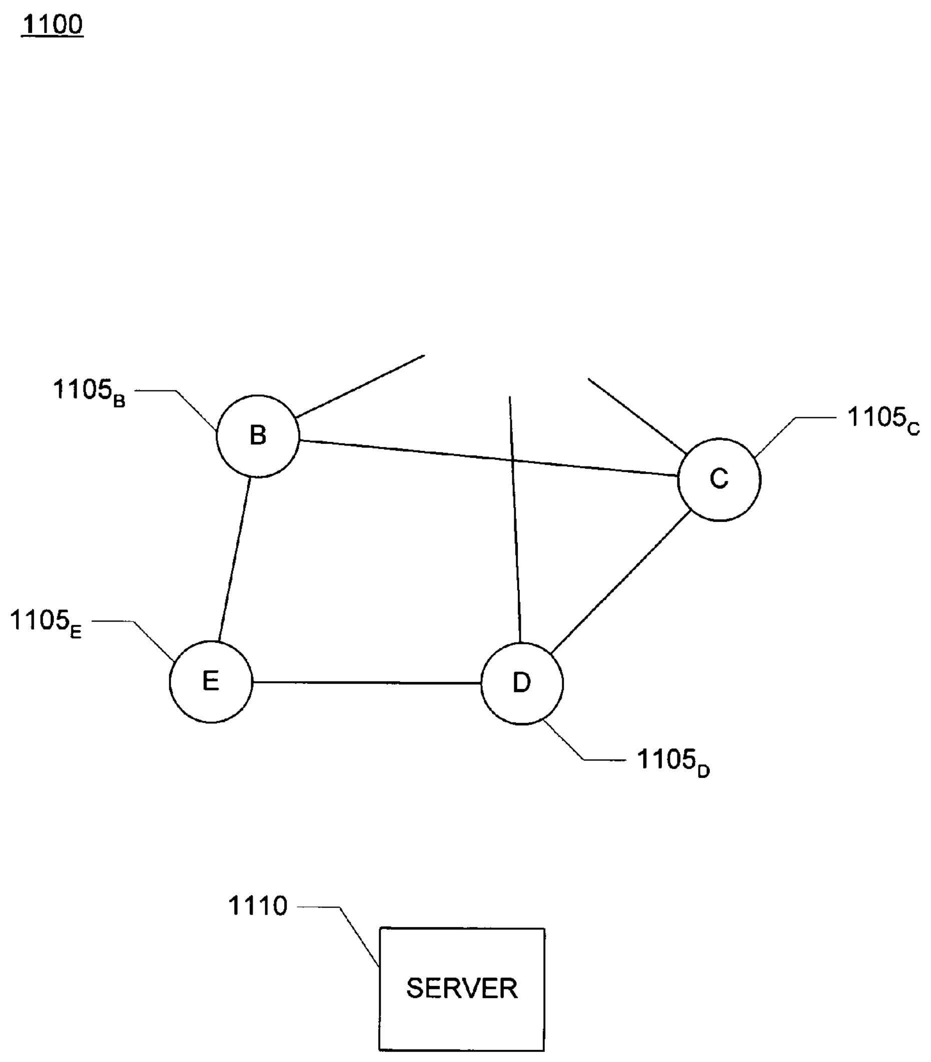


FIG. 16

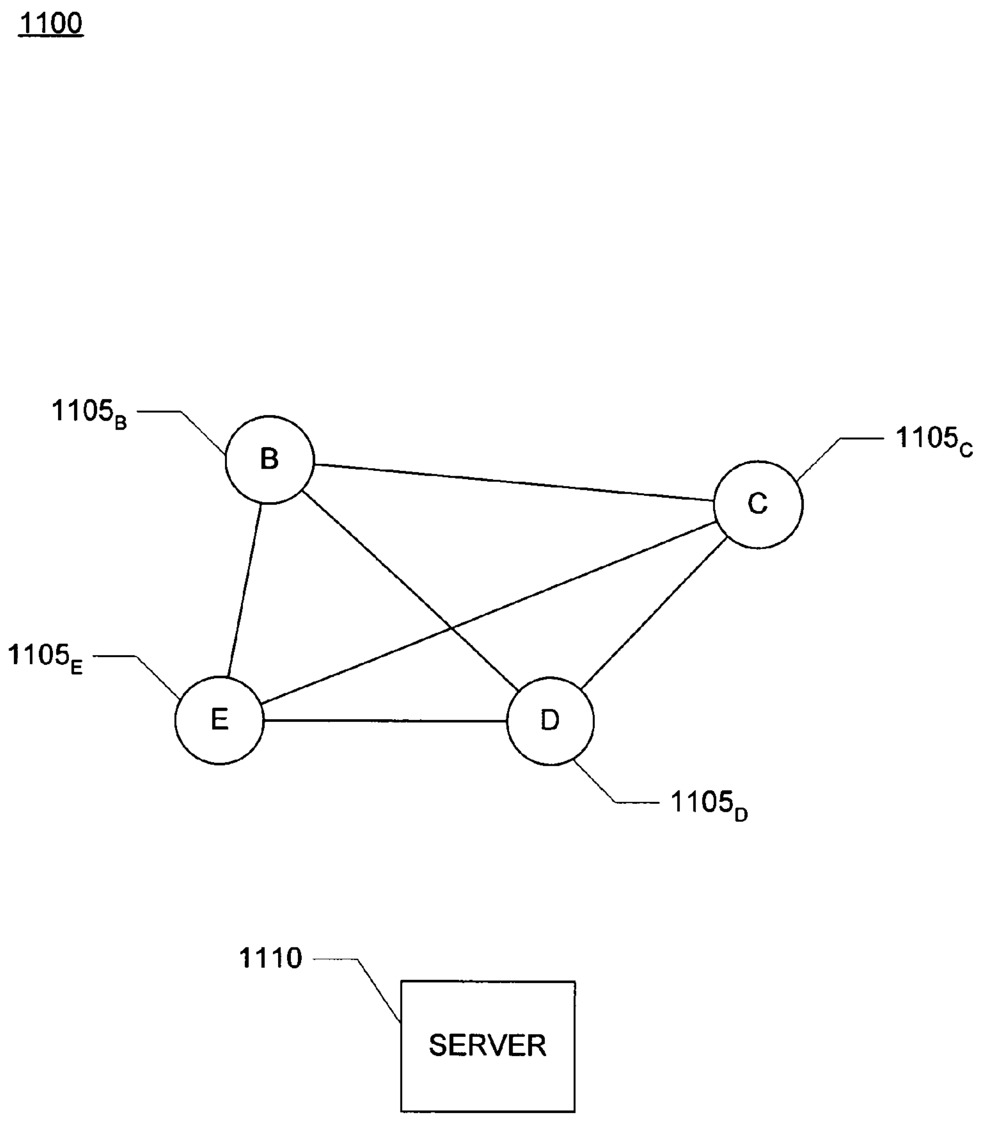


FIG. 17

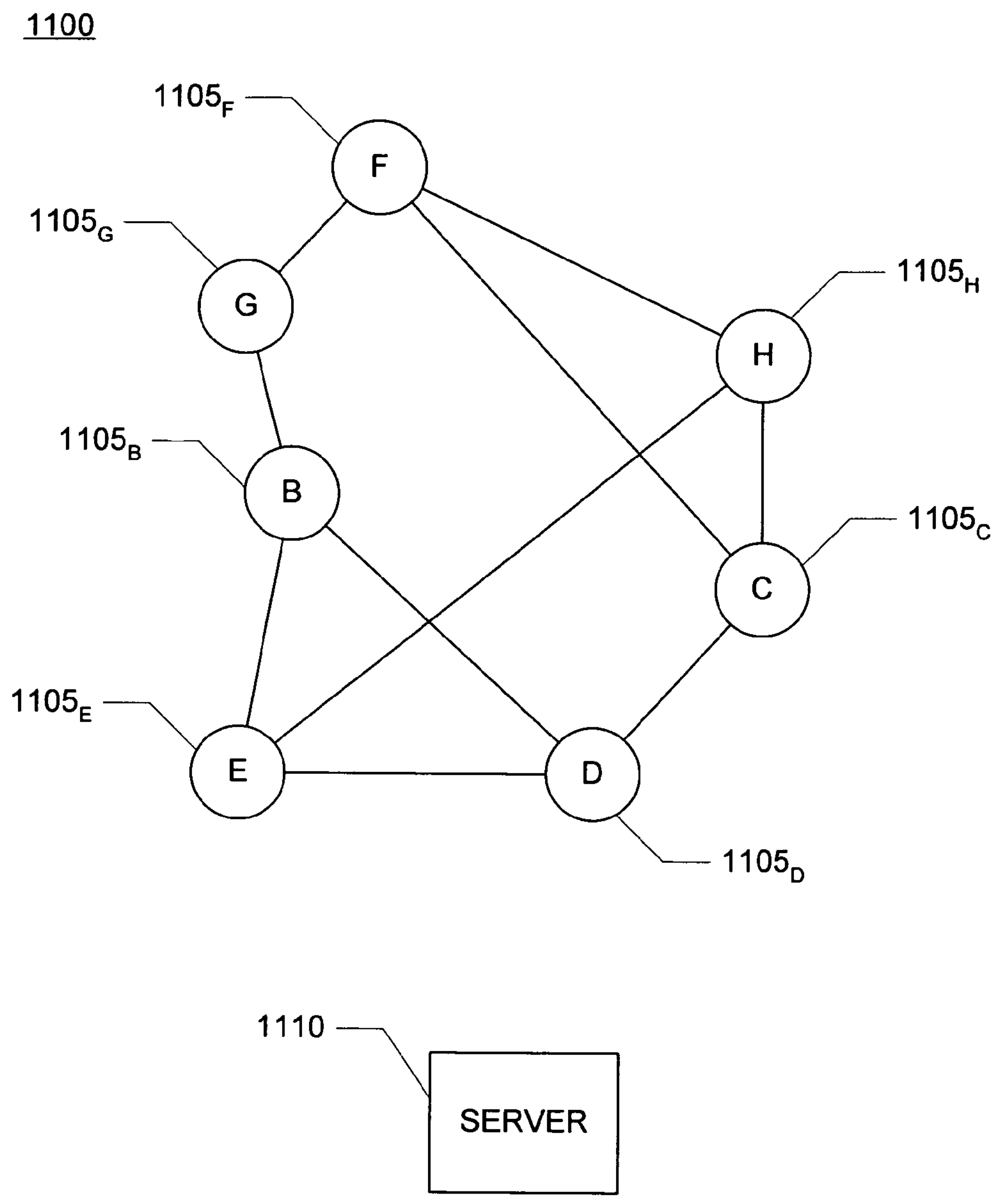

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 18

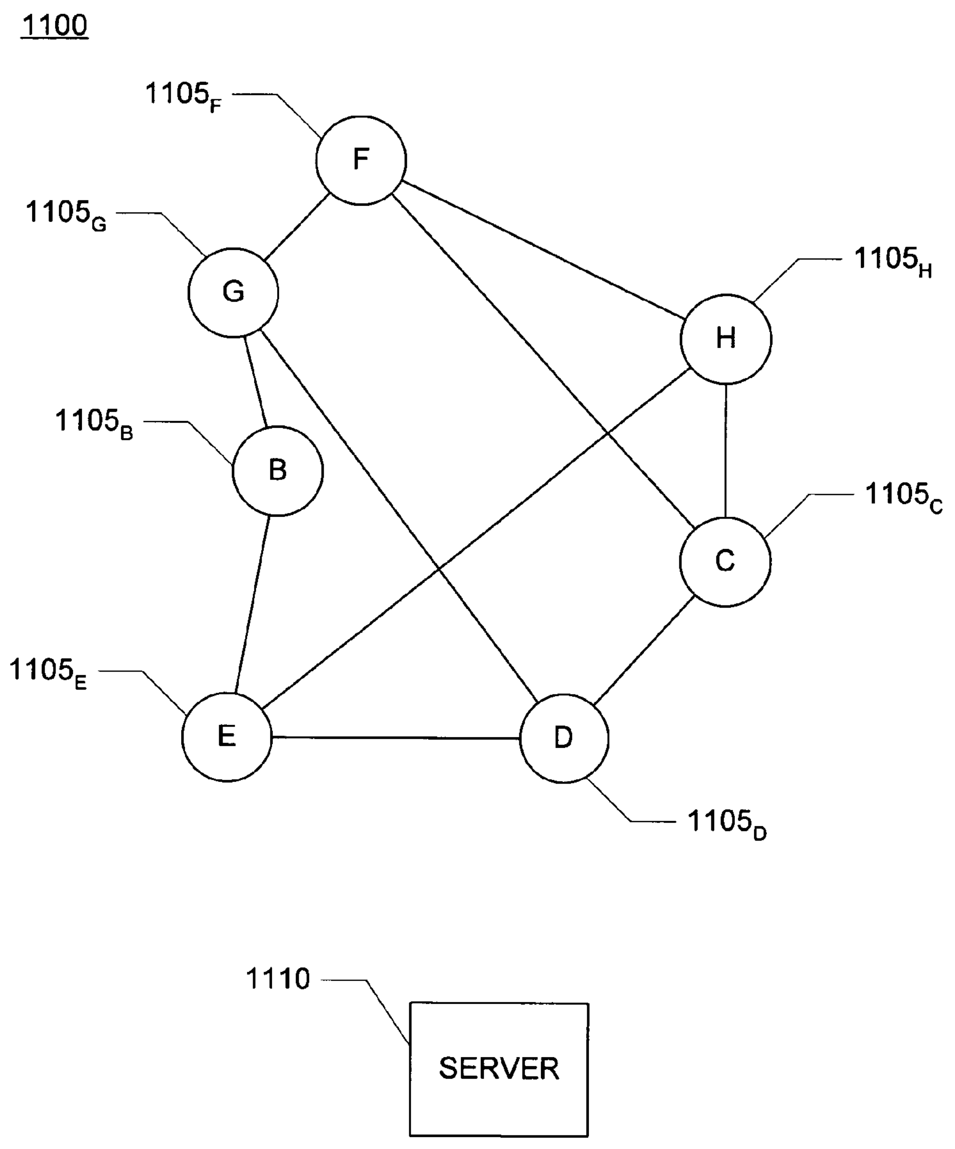

1100
1105_F
1105_G
1105_H
1105_B
1105_C
1105_E
1105_D
F
G
H
B
C
E
D
1110
SERVER

FIG. 19

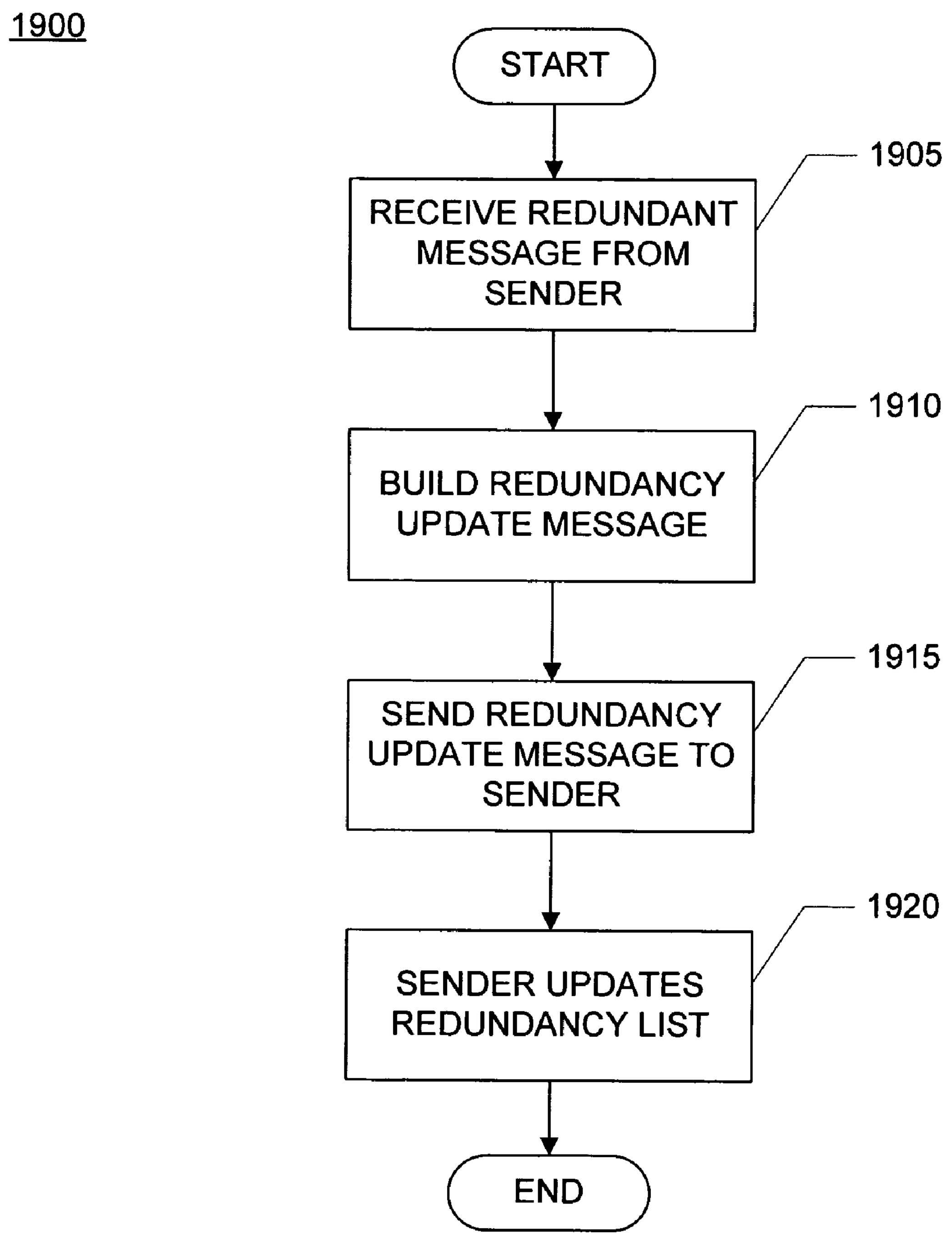

1900
START
1905
RECEIVE REDUNDANT MESSAGE FROM SENDER
1910
BUILD REDUNDANCY UPDATE MESSAGE
1915
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1920
SENDER UPDATES REDUNDANCY LIST
END

FIG. 20

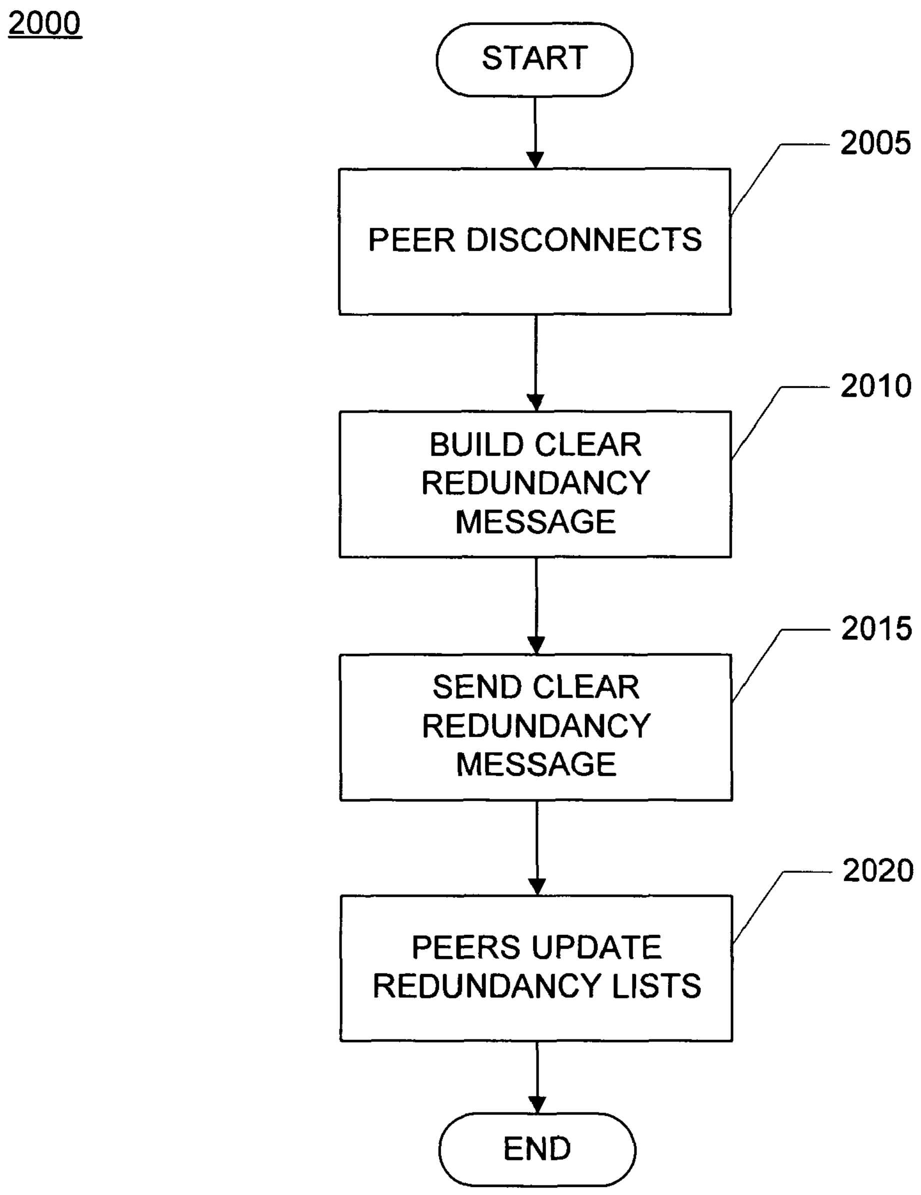

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

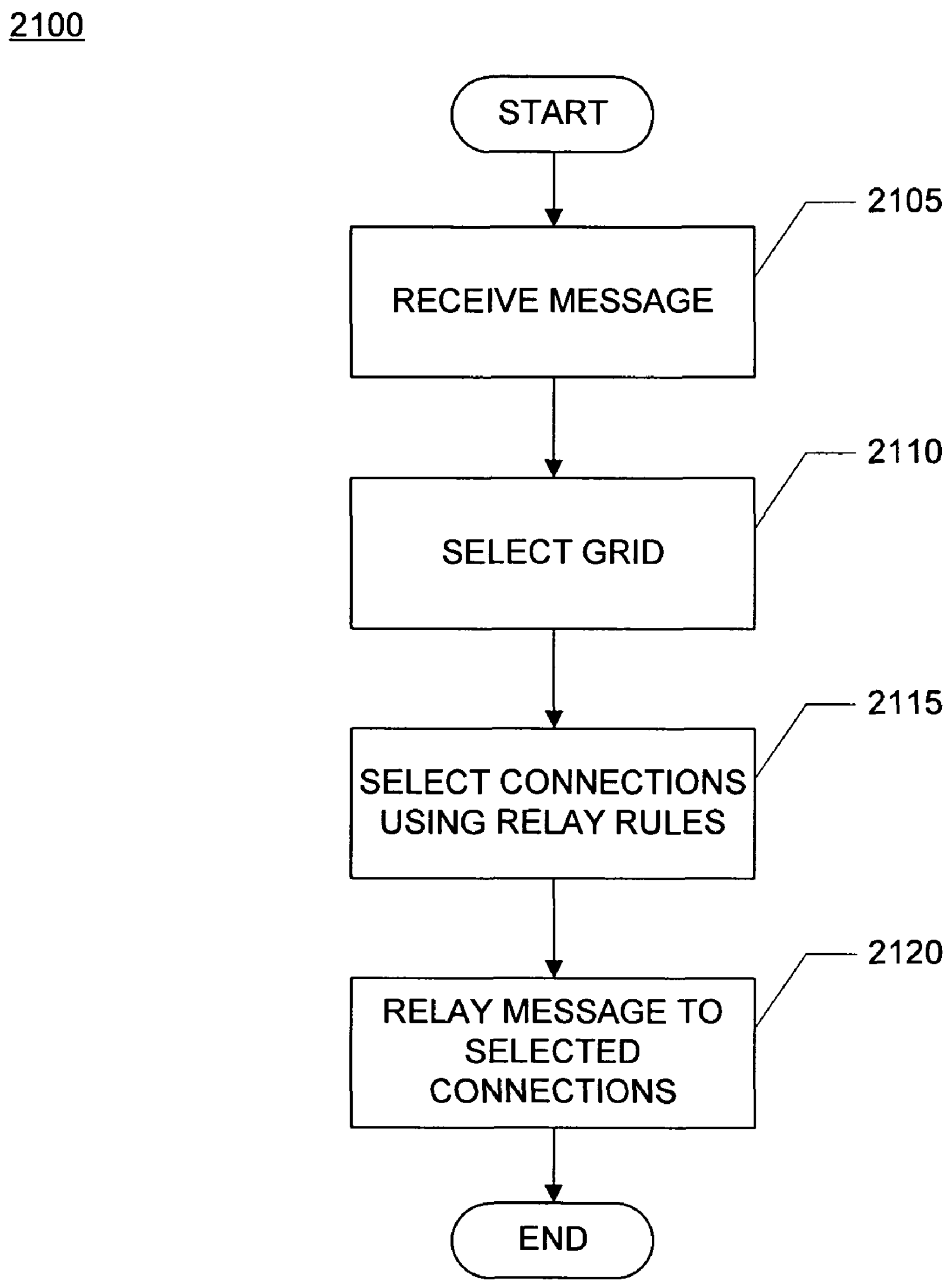

2100
START
2105
RECEIVE MESSAGE
2110
SELECT GRID
2115
SELECT CONNECTIONS USING RELAY RULES
2120
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 22

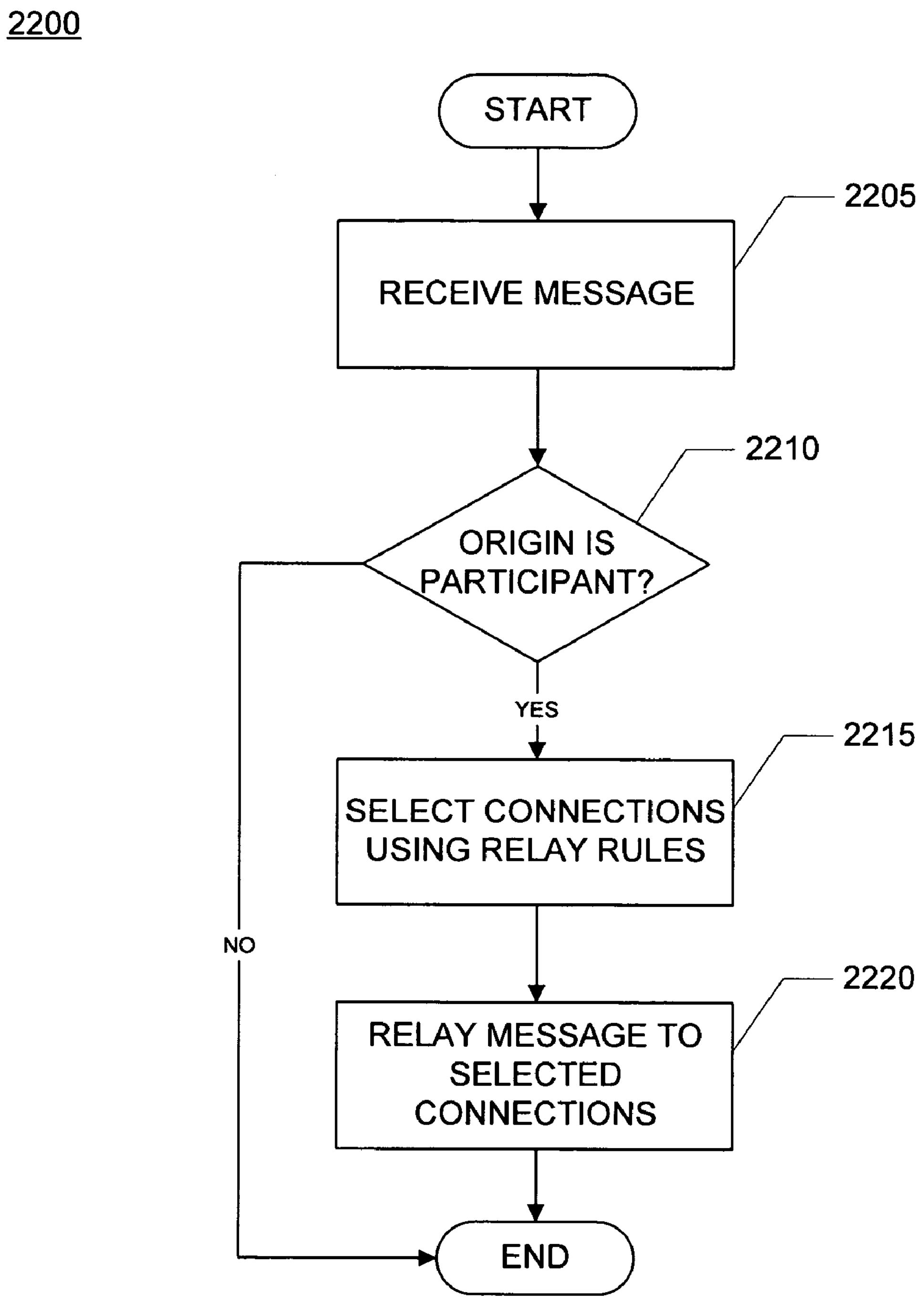

2200
START
RECEIVE MESSAGE
2205
ORIGIN IS PARTICIPANT?
2210
YES
NO
SELECT CONNECTIONS USING RELAY RULES
2215
RELAY MESSAGE TO SELECTED CONNECTIONS
2220
END

FIG. 23

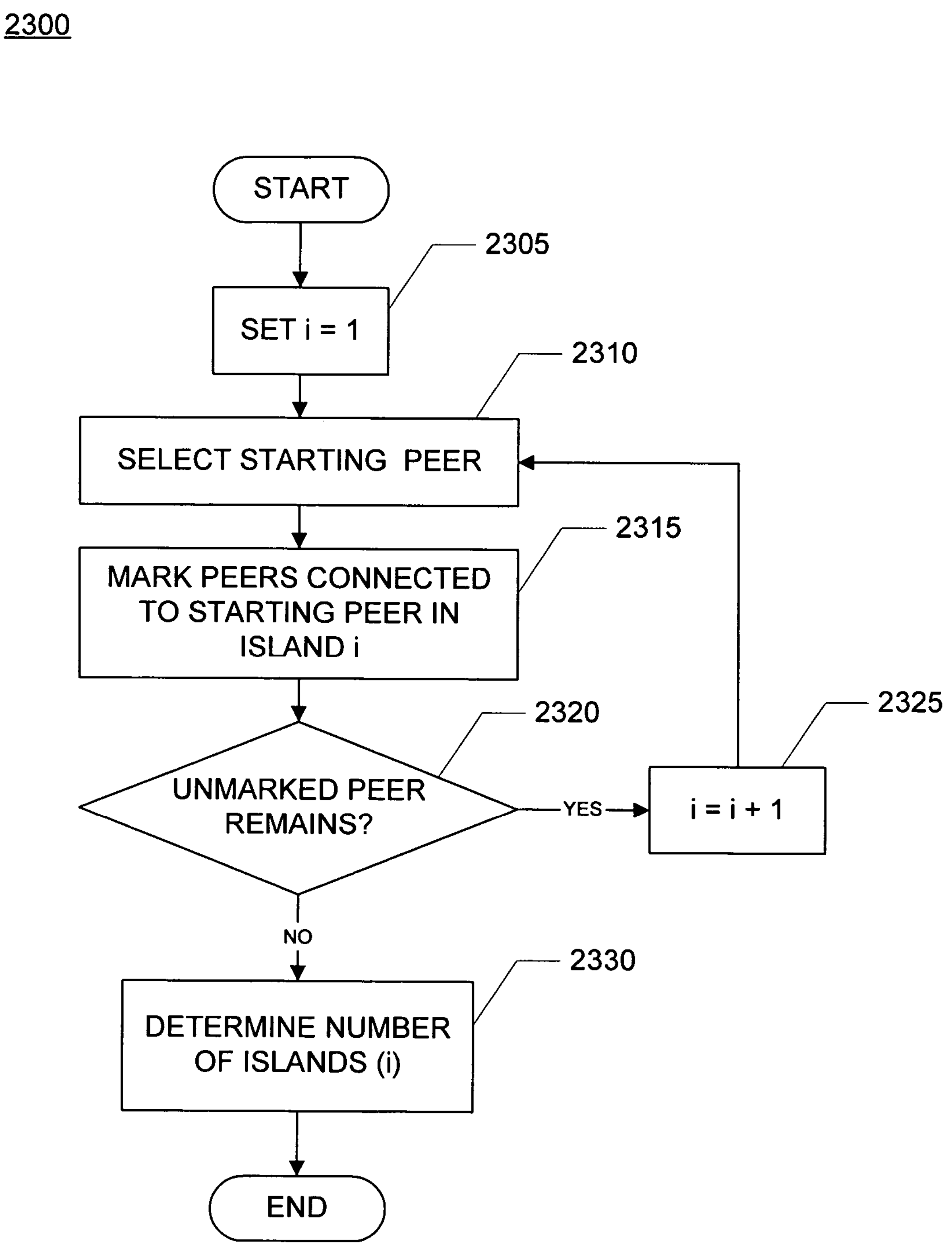

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

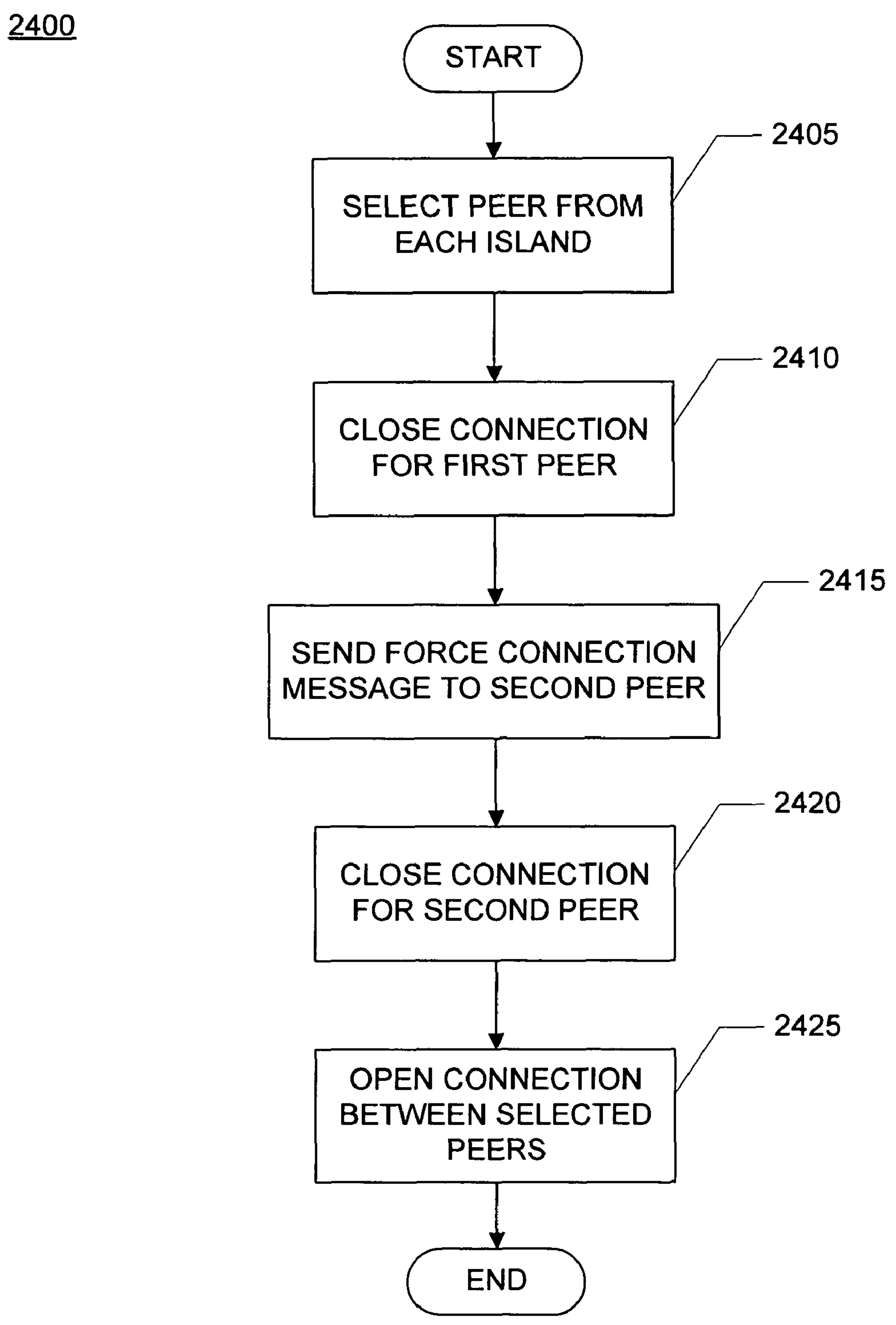

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

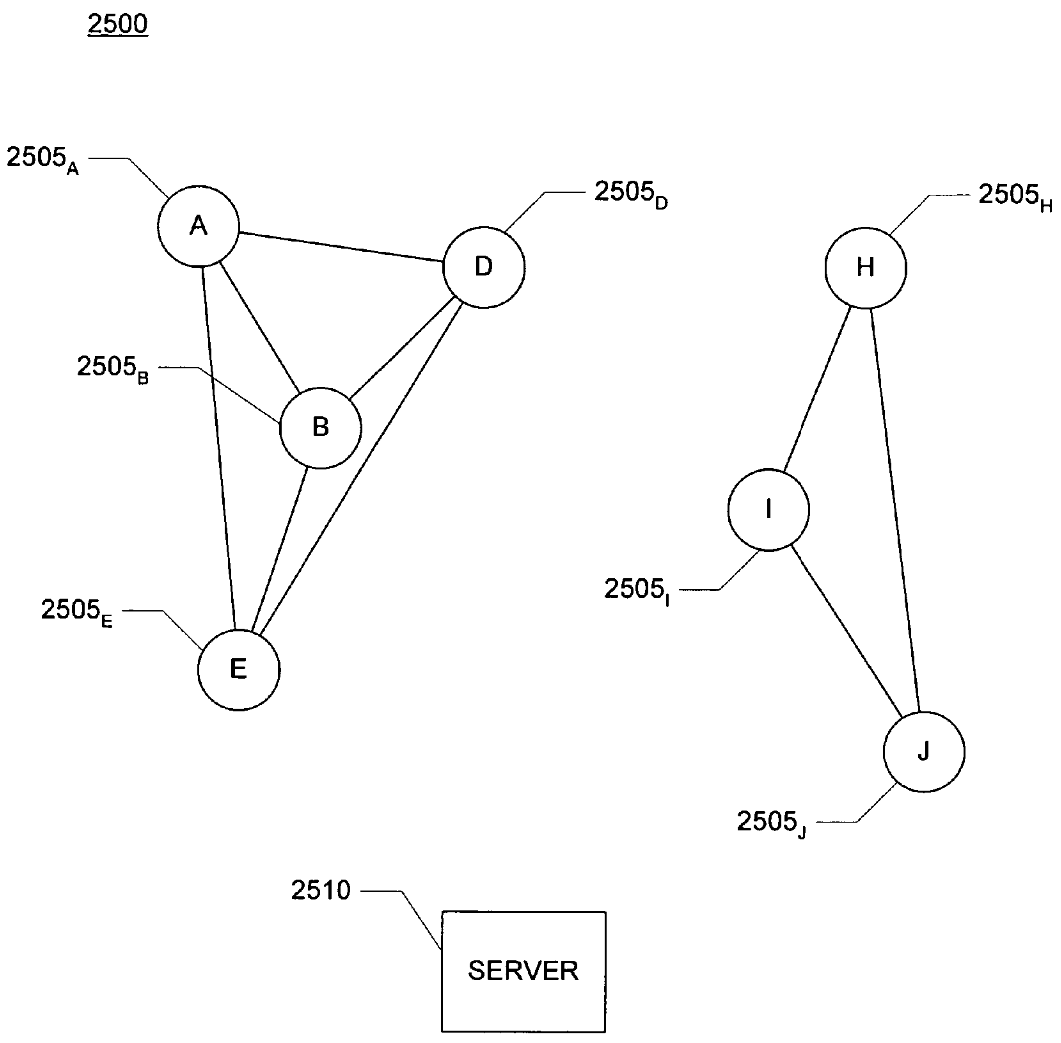

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 26

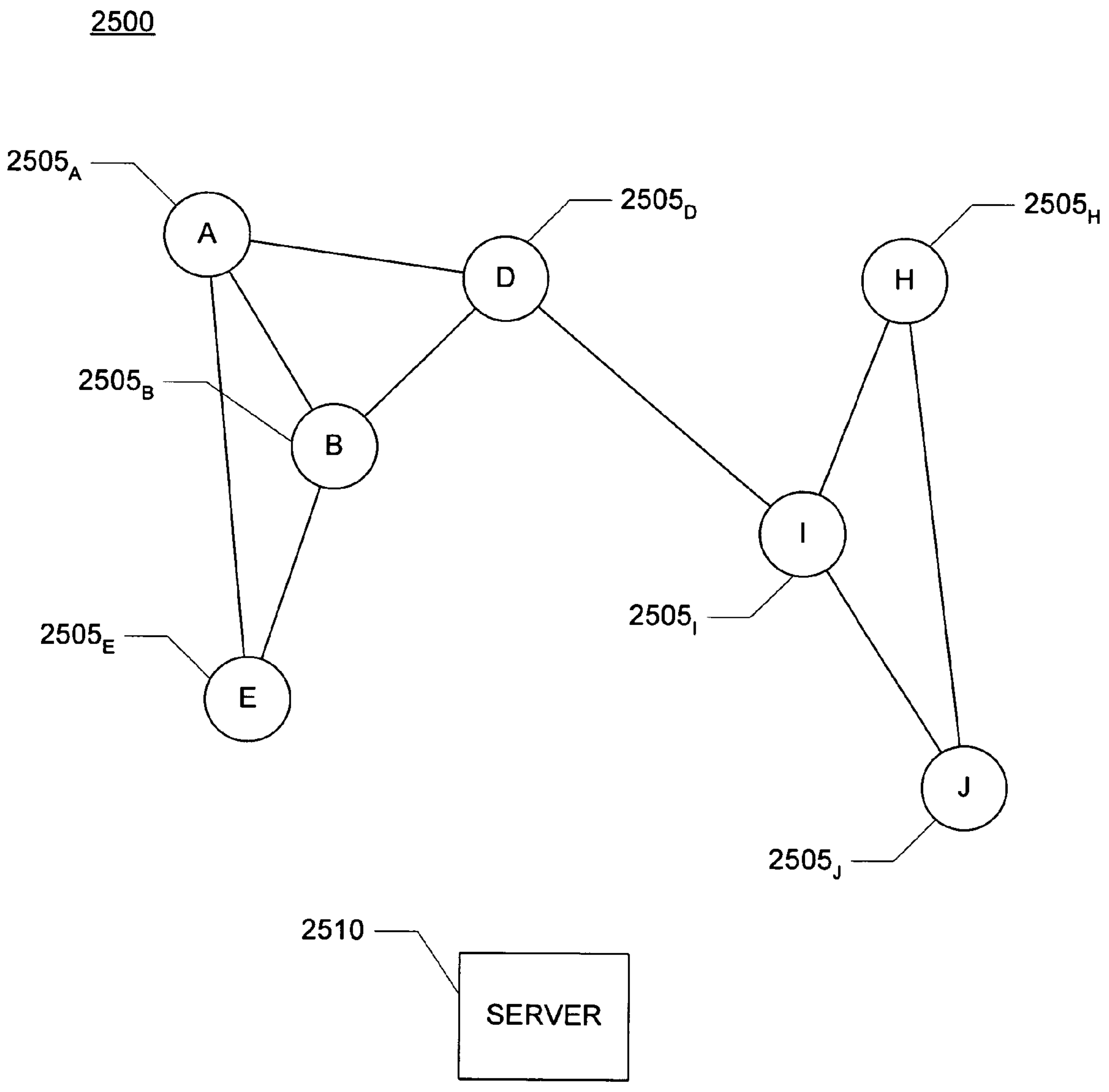

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 27

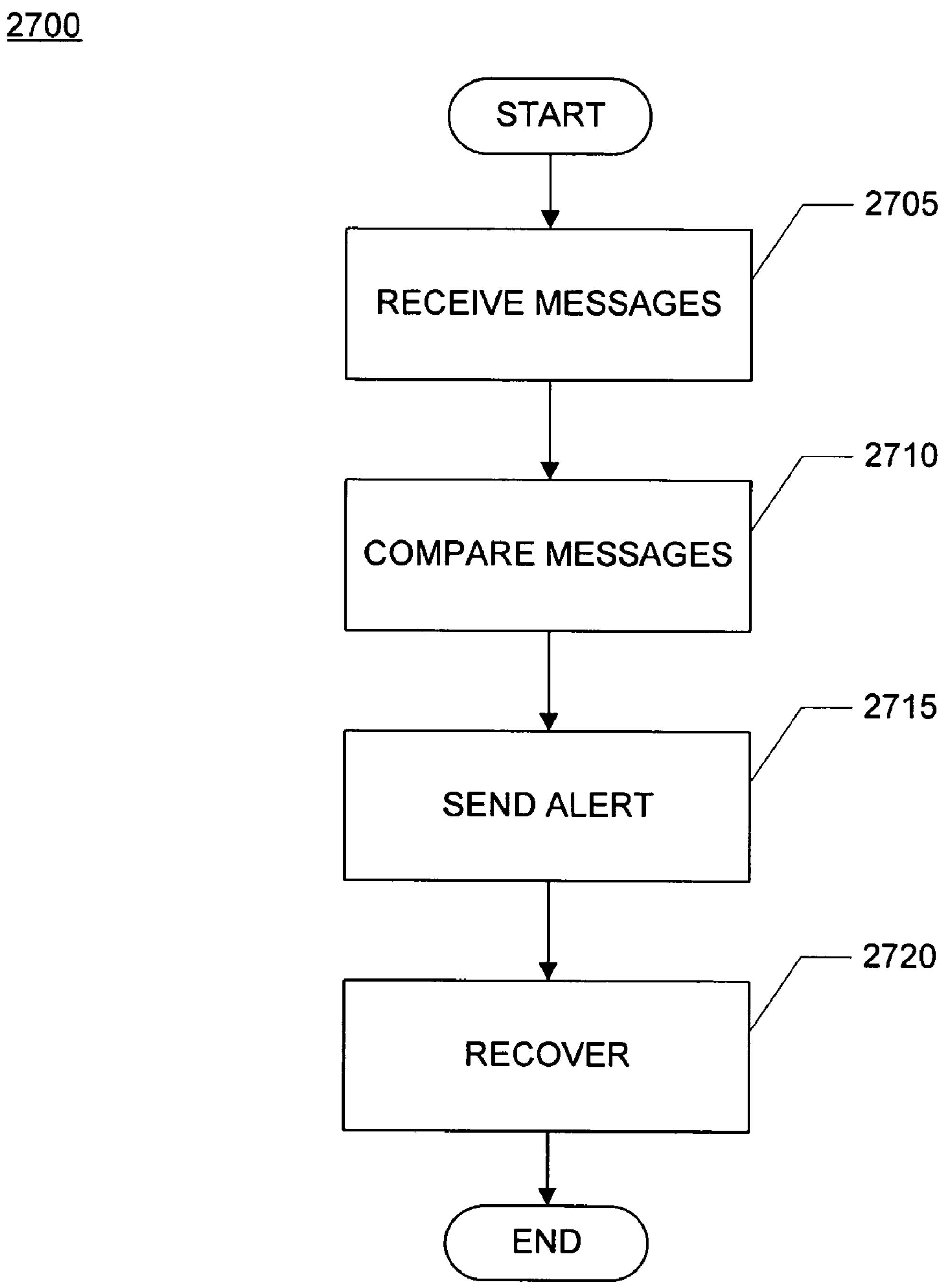

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

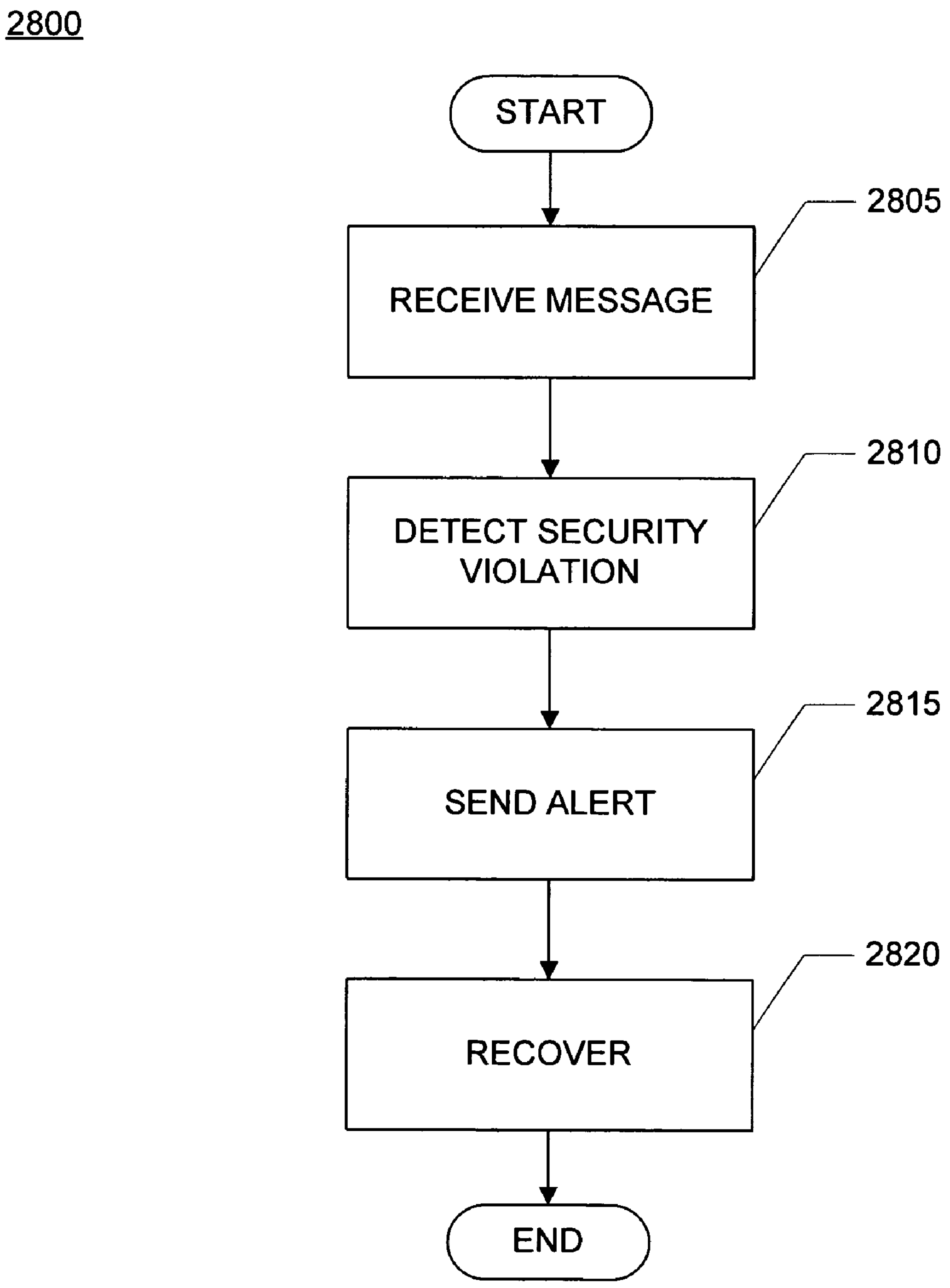

2800
START
2805
RECEIVE MESSAGE
2810
DETECT SECURITY VIOLATION
2815
SEND ALERT
2820
RECOVER
END

FIG. 29

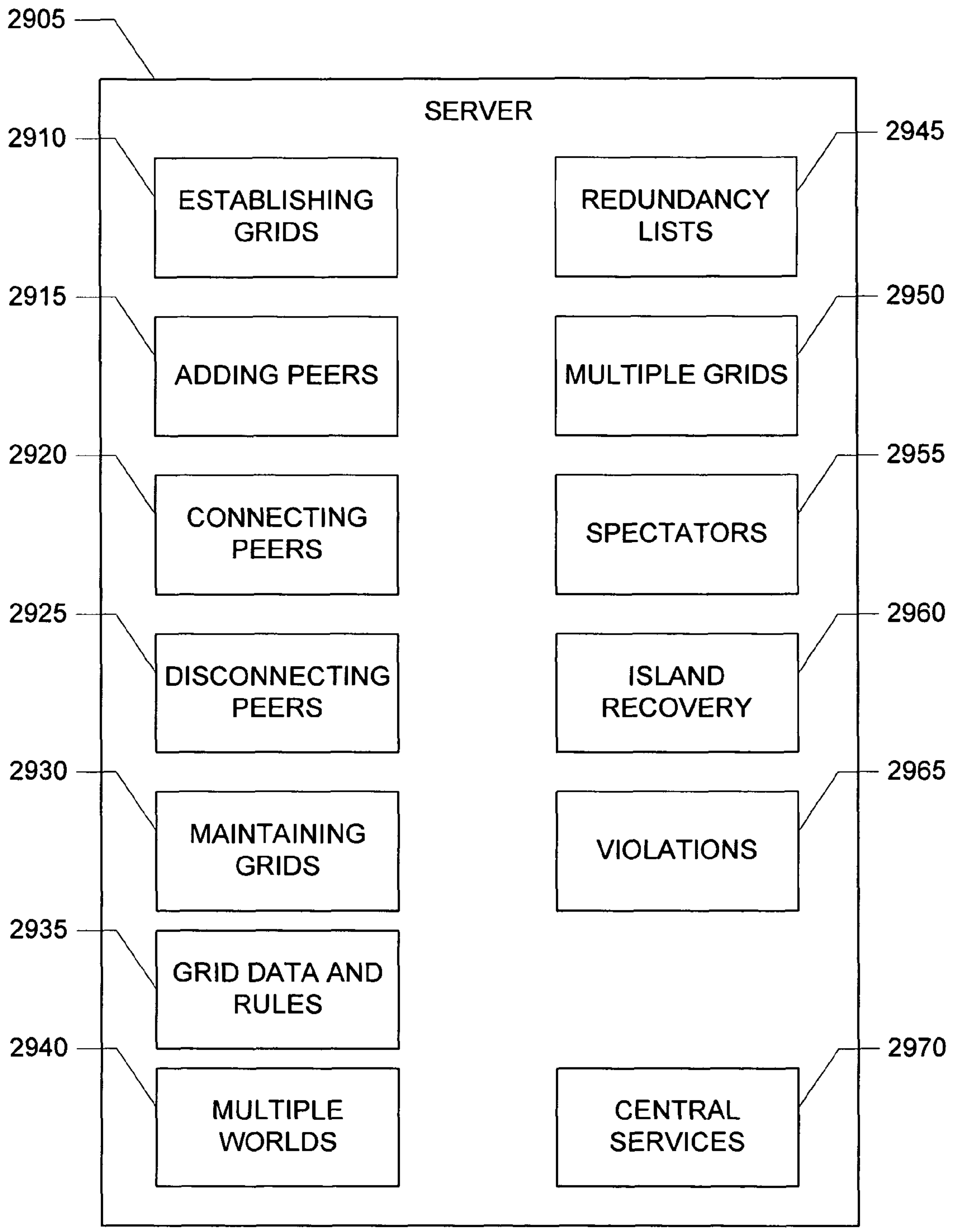

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

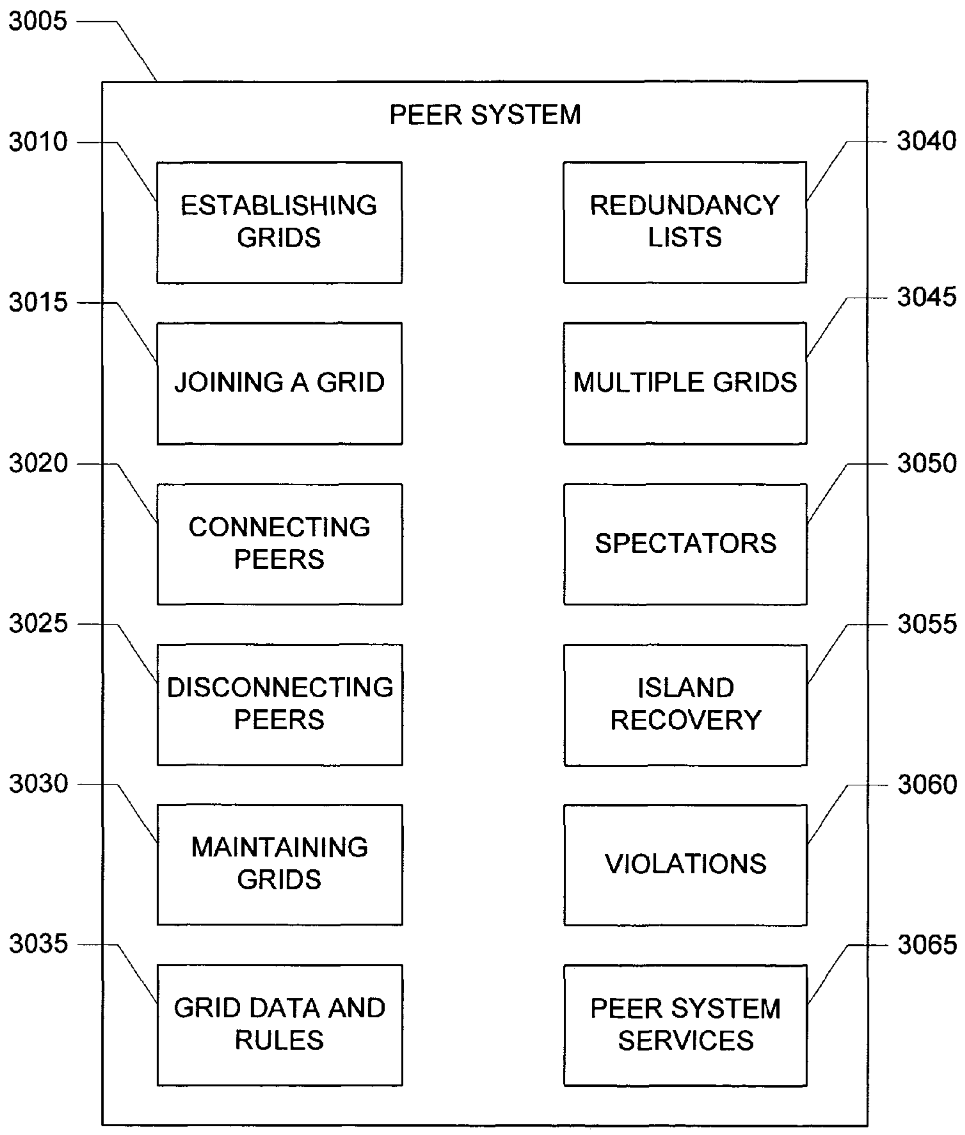

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

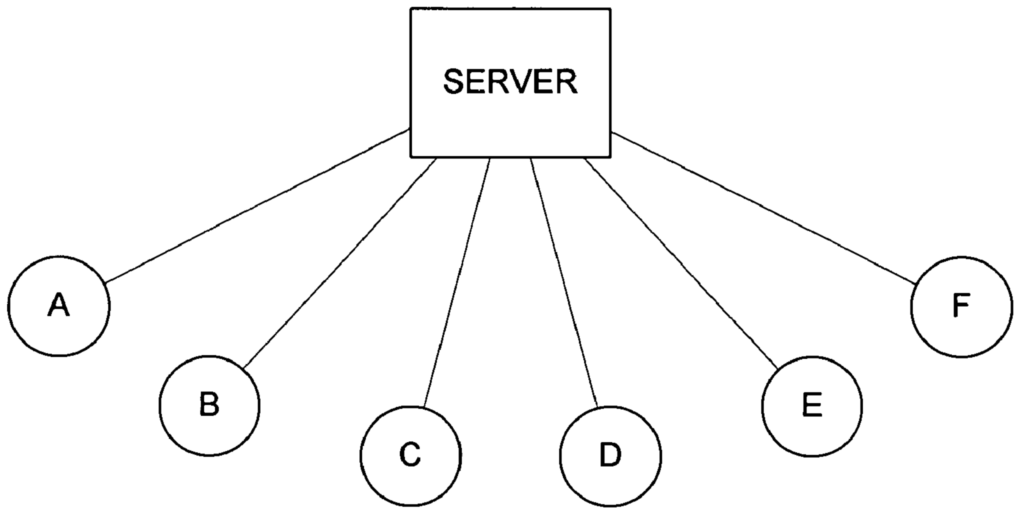
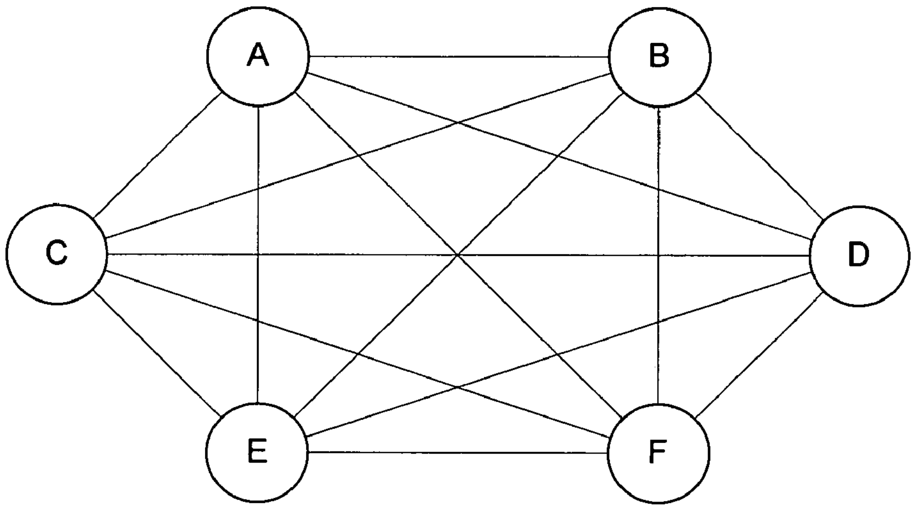


FIG. 31B



MULTIPLE PEER-TO-PEER RELAY NETWORKS

This application claims the benefit of U.S. Provisional Application No. 60/513,098 ("PEER-TO-PEER RELAY NETWORK"), filed Oct. 20, 2003, the disclosure of which is incorporated herein by reference.

This application is related to the U.S. applications Ser. No. 10/700,798, filed on Nov. 3, 2003, Ser. No. 10/701,014, filed on Nov. 3, 2003, Ser. No. 10/700,777, filed on Nov. 3, 2003, Ser. No. 10/701,298, filed on Nov. 3, 2003, Land Ser. No. 10/700,797, filed on Nov. 3, 2003.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N-1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a network environment supporting multiple peer-to-peer relay networks includes: a first peer-to-peer relay network including N**1** peer systems; and a second peer-to-peer relay network including N2 peer systems; wherein each peer system in said first peer-to-peer relay network is connected to a number of other peer systems in said first peer-to-peer relay network that is less than or equal to a first connection limit, said first connection limit is greater than or equal to 2, said first connection limit is less than or equal to N1-2, each peer system in said first peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a first set of one or more relay rules, each peer system in said second peer-to-peer relay network is connected to a number of other peer systems in said second peer-to-peer relay network that is less than or equal to a second connection limit, said second connection limit is greater than or equal to 2, said second connection limit is less than or equal to N2-2, each peer system in said second peer-to-peer relay network is configured to relay data to peer systems connected to that peer system according to a second set of one or more relay rules, and at least one peer system in said first peer-to-peer relay network is also in said second peer-to-peer relay network.

In one implementation, a method of relaying data in a peer-to-peer relay network includes: receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network; selecting a peer-to-peer relay network corresponding to said received data, wherein said selected peer-to-peer relay network has a corresponding set of one or more relay rules; applying said set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and relaying said data to any peer systems selected by applying said set of one or more relay rules.

In one implementation, a peer system in a peer-to-peer relay network includes: means for receiving data at a relaying peer system from a sending peer system connected to the relaying peer system in a peer-to-peer relay network; means for selecting a peer-to-peer relay network corresponding to said received data, wherein said selected peer-to-peer relay network has a corresponding set of one or more relay rules; means for applying said set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and means for relaying said data to any peer systems selected by applying said set of one or more relay rules.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. Patent Applications Nos. 10/211,075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed 31 Jul. 2002 (now U.S. Pat. No. 7,421,471), and 10/359,359 ("Multi-User Application Programming Interface"), filed 4 Feb. 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of 3. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system **105**$_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3 other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**.

A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members, not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet).

In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins, peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server for the grid of the changing connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively; one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A network environment supporting multiple peer-to-peer relay networks, comprising:

a main peer-to-peer relay network including all peer systems in the multiple peer-to-peer relay networks, at least one of the peer systems including at least one processor, the main peer-to-peer network having sub-networks within the main peer-to-peer relay network, wherein each peer system of a sub-network is also a member of the main peer-to-peer relay network;

a first peer-to-peer relay network including a plurality of first peer systems that are a first sub-network of the main peer-to-peer relay network, at least one of said first peer systems including at least one processor, and including a first particular peer system and a second particular peer system;

a second peer-to-peer relay network including a plurality of second peer systems that are a second sub-network of the main peer-to-peer network, at least one of said second peer systems including at least one processor, and including the first particular peer system and the second particular peer system; and

wherein the first particular peer system has a connection to the second particular peer in the first peer-to-peer relay network and the first particular peer system does not have a connection to the second particular peer in the second peer-to-peer relay network, and

wherein a message addressed from a peer in the first peer-to-peer relay network to another peer in the first peer-to-peer relay network is relayed only to peers in the first peer-to-peer relay network, and

wherein a message addressed from a peer in the first peer-to-peer relay network to a peer in the main peer-to-peer relay network before the first sub-network is established is relayed to all peers in the main peer-to-peer relay network.

2. The network environment of claim **1**, further comprising:

a server connected to each peer system.

3. The network environment of claim **1**, wherein:

the peer systems in said first peer-to-peer relay network represent players in an online game.

4. The network environment of claim **3**, wherein:

the peer systems in said first peer-to-peer relay network represent players in said online game that are on the same team.

5. The network environment of claim **1**, wherein:

data relayed in said first peer-to-peer relay network is network service data.

6. The network environment of claim **1**, wherein:

data relayed in said first peer-to-peer relay network is data for an online environment.

7. The network environment of claim **6**, wherein:

data relayed in said first peer-to-peer relay network is data for a lobby environment.

8. The network environment of claim **7**, wherein:

data relayed in said first peer-to-peer relay network is data for a chat room in said lobby environment.

9. The network environment of claim **6**, wherein:

data relayed in said second peer-to-peer relay network is data for an online game.

10. The network environment of claim **1**, wherein:

at least one peer system is a network-enabled game console.

11. The network environment of claim **1**, wherein:

at least two peer systems are connected through the Internet.

12. A method of relaying data in a peer-to-peer relay network, comprising:

establishing a main peer-to-peer relay network including all peer systems in the peer-to-peer relay network, at least one of the peer systems including at least one processor, the main peer-to-peer network having sub-networks within the main peer-to-peer relay network, wherein each peer system of a sub-network is also a member of the main peer-to-peer relay network;

establishing a first peer-to-peer relay network including a plurality of first peer systems that are a first sub-network of the main peer-to-peer relay network, at least one of said first peer systems including at least one processor, and including a first particular peer system and a second particular peer system;

establishing a second peer-to-peer relay network including a plurality of second peer systems that are a second sub-network of the main peer-to-peer network, at least one of said second peer systems including at least one processor, and including the first particular peer system and the second particular peer system;

wherein the first particular peer system has a connection to the second particular peer in the first peer-to-peer relay network and the first particular peer system does not have a connection to the second particular peer in the second peer-to-peer relay network;

receiving data at a relaying peer system in the first peer-to-peer relay network from a sending peer system connected to the relaying peer system;

selecting another peer in the first peer-to-peer relay network corresponding to said received data; and

relaying said data to the another peer system,

wherein a message addressed from a peer in the first peer-to-peer relay network to another peer in the first peer-to-peer relay network is relayed only to peers in the first peer-to-peer relay network, and

wherein a message addressed from a peer in the first peer-to-peer relay network to a peer in the main peer-to-peer relay network before the first sub-network is established is relayed to all peers in the main peer-to-peer relay network.

13. The method of claim **12**, wherein:

said relaying peer system is in two or more peer-to-peer relay networks, and said relaying peer system has respective sets of one or more connections to other peer systems for each peer-to-peer relay network to which said relaying peer system belongs.

14. The method of claim **12**, wherein:

said relaying peer system stores a respective connection limit and a respective set of one of more relay rules for each peer-to-peer relay network to which said relaying peer system belongs, a connection limit defines a number of other peer systems up to which a peer system is permitted to connect in that peer-to-peer relay network, and a set of one or more relay rules defines how a peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

15. A peer system in a peer-to-peer relay network, comprising:

means for establishing a main peer-to-peer relay network including all peer systems in the peer-to-peer relay network, at least one of the peer systems including at least one processor, the main peer-to-peer network having sub-networks within the main peer-to-peer relay network, wherein each peer system of a sub-network is also a member of the main peer-to-peer relay network;

means for establishing a first peer-to-peer relay network including a plurality of first peer systems that are a first sub-network of the main peer-to-peer relay network, at least one of said first peer systems including at least one processor, and including a first particular peer system and a second particular peer system;

means for establishing a second peer-to-peer relay network including a plurality of second peer systems that are a

second sub-network of the main peer-to-peer network, at least one of said second peer systems including at least one processor, and including the first particular peer system and the second particular peer system;

wherein the first particular peer system has a connection to the second particular peer in the first peer-to-peer relay network and the first particular peer system does not have a connection to the second particular peer in the second peer-to-peer relay network;

means for receiving data at a relaying peer system in the first peer-to-peer relay network from a sending peer system connected to the relaying peer system;

means for selecting another peer in the first peer-to-peer relay network corresponding to said received data; and

means for relaying said data to the another peer system,

wherein a message addressed from a peer in the first peer-to-peer relay network to another peer in the first peer-to-peer relay network is relayed only to peers in the first peer-to-peer relay network, and

wherein a message addressed from a peer in the first peer-to-peer relay network to a peer in the main peer-to-peer relay network before the first sub-network is established is relayed to all peers in the main peer-to-peer relay network.

16. The peer system of claim **15**, wherein:

said peer system is in two or more peer-to-peer relay networks, and said peer system has respective sets of one or more connections to other peer systems for each peer-to-peer relay network to which said peer system belongs.

17. The peer system of claim **15**, wherein:

said peer system stores a respective connection limit and a respective set of one of more relay rules for each peer-to-peer relay network to which said peer system belongs, a connection limit defines a number of other peer systems up to which a peer system is permitted to connect in that peer-to-peer relay network, and a set of one or more relay rules defines how a peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

18. A non-transitory computer-readable storage medium having a computer-readable program embodied therein, said computer readable program adapted to be executed to implement a peer system in a peer-to-peer relay network, the method comprising:

establishing a main peer-to-peer relay network including all peer systems in the peer-to-peer relay network, at least one of the peer systems including at least one processor, the main peer-to-peer network having sub-networks within the main peer-to-peer relay network, wherein each peer system of a sub-network is also a member of the main peer-to-peer relay network;

establishing a first peer-to-peer relay network including a plurality of first peer systems that are a first sub-network of the main peer-to-peer relay network, at least one of said first peer systems including at least one processor, and including a first particular peer system and a second particular peer system;

establishing a second peer-to-peer relay network including a plurality of second peer systems that are a second sub-network of the main peer-to-peer network, at least one of said second peer systems including at least one processor, and including the first particular peer system and the second particular peer system;

wherein the first particular peer system has a connection to the second particular peer in the first peer-to-peer relay network and the first particular peer system does not have a connection to the second particular peer in the second peer-to-peer relay network;

receiving data at a relaying peer system in the first peer-to-peer relay network from a sending peer system connected to the relaying peer system;

selecting another peer in the first peer-to-peer relay network corresponding to said received data; and

relaying said data to the another peer system,

wherein a message addressed from a peer in the first peer-to-peer relay network to another peer in the first peer-to-peer relay network is relayed only to peers in the first peer-to-peer relay network, and

wherein a message addressed from a peer in the first peer-to-peer relay network to a peer in the main peer-to-peer relay network before the first sub-network is established is relayed to all peers in the main peer-to-peer relay network, and

wherein each peer independently maintains a list of available networks and a list of peers in each network.

19. The non-transitory computer-readable storage medium of claim **18**, wherein:

said peer system is in two or more peer-to-peer relay networks, and said peer system has respective sets of one or more connections to other peer systems for each peer-to-peer relay network to which said peer system belongs.

20. The non-transitory computer-readable storage medium of claim **18**, wherein:

said peer system stores a respective connection limit and a respective set of one of more relay rules for each peer-to-peer relay network to which said peer system belongs, a connection limit defines a number of other peer systems up to which a peer system is permitted to connect in that peer-to-peer relay network, and a set of one or more relay rules defines how a peer system is to relay data to other peer systems connected to that peer system in that peer-to-peer relay network.

* * * * *

EXHIBIT 93

US008396984B2

(12) **United States Patent**
Van Datta et al.

(10) **Patent No.: US 8,396,984 B2**
(45) **Date of Patent: Mar. 12, 2013**

(54) **PEER-TO-PEER RELAY NETWORK WITH DECENTRALIZED CONTROL**

(75) Inventors: **Glen Van Datta**, San Diego, CA (US); **Anthony Mai**, San Marcos, CA (US)

(73) Assignee: **Sony Computer Entertainment America Inc.**, Foster City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 272 days.

(21) Appl. No.: **11/895,978**

(22) Filed: **Aug. 28, 2007**

(65) **Prior Publication Data**

US 2008/0046555 A1 Feb. 21, 2008

Related U.S. Application Data

(62) Division of application No. 10/700,798, filed on Nov. 3, 2003, now Pat. No. 7,627,678.

(60) Provisional application No. 60/513,098, filed on Oct. 20, 2003.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/238**; 709/203; 709/223; 709/243

(58) **Field of Classification Search** 709/223, 709/225, 227, 238, 204, 243, 203, 217; 370/389, 370/401
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,787,051	A	11/1988	Olson	
4,843,568	A	6/1989	Krueger	
5,128,671	A	7/1992	Thomas, Jr.	
5,528,265	A	6/1996	Harrison	
5,624,316	A *	4/1997	Roskowski et al.	463/45
5,701,427	A	12/1997	Lathrop	
6,157,368	A	12/2000	Fager	
6,269,099	B1 *	7/2001	Borella et al.	370/389
6,375,572	B1	4/2002	Masuyama	
6,421,347	B1 *	7/2002	Borgstahl et al.	370/310
6,438,603	B1 *	8/2002	Ogus	709/238
6,446,055	B1 *	9/2002	Grand	706/10
6,701,344	B1	3/2004	Holt et al.	
7,065,579	B2 *	6/2006	Traversat et al.	709/225
7,130,921	B2 *	10/2006	Goodman et al.	709/238
7,174,382	B2 *	2/2007	Ramanathan et al.	709/227
7,177,950	B2	2/2007	Narayan et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0 913 965	5/1999
EP	1 107 508	6/2001
WO	WO 02/11366	2/2002
WO	WO 03/069495	8/2003

OTHER PUBLICATIONS

Steven Hessing: "Peer to Peer Messaging Protocol (PPMP)" Internet Draft, Apr. 2002, pp. 1-57, XP015001173.

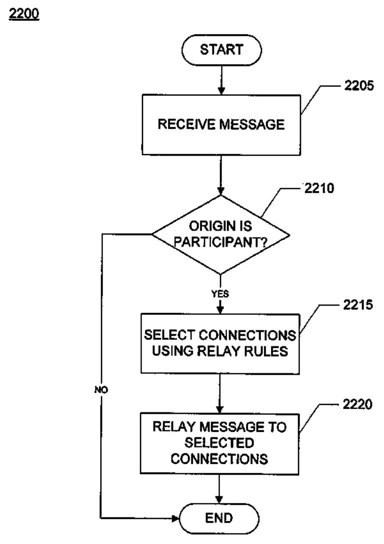
(Continued)

Primary Examiner — Ramy M Osman
(74) *Attorney, Agent, or Firm* — Frommer Lawrence & Haug LLP; William S. Frommer; Paul A. Levy

(57) **ABSTRACT**

Methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N-2, and each peer system is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules.

9 Claims, 31 Drawing Sheets


2200
START
RECEIVE MESSAGE
2205
ORIGIN IS PARTICIPANT?
2210
YES
NO
SELECT CONNECTIONS USING RELAY RULES
2215
RELAY MESSAGE TO SELECTED CONNECTIONS
2220
END

U.S. PATENT DOCUMENTS

7,263,070	B1 *	8/2007	Delker et al.	709/227
7,272,636	B2 *	9/2007	Pabla	709/223
2001/0044339	A1	11/2001	Cordero et al.	
2002/0055989	A1	5/2002	Stringer-Calvert et al.	
2002/0085097	A1	7/2002	Colmenarez et al.	
2002/0107786	A1 *	8/2002	Lehmann-Haupt et al.	705/37
2002/0119821	A1	8/2002	Sen et al.	
2002/0143855	A1 *	10/2002	Traversat et al.	709/202
2002/0156875	A1 *	10/2002	Pabla	709/220
2002/0161821	A1 *	10/2002	Narayan et al.	709/204
2002/0184310	A1	12/2002	Traversat et al.	
2003/0055892	A1	3/2003	Huitema et al.	
2004/0103179	A1	5/2004	Damm et al.	
2004/0212589	A1	10/2004	Hall et al.	
2005/0063409	A1	3/2005	Oommen	
2005/0080858	A1	4/2005	Pessach	

OTHER PUBLICATIONS

Song Jiang et al: “FloodTrial : an efficient file search technique in unstructured peer-to-peer systems” GLOBECOM 2003, vol. 5, Dec. 1, 2003, pp. 2891-2895, XP010678188.

Dutkiewicz E Ed—Institute of Electrical and Electronics Engineers: “Impact of transmit range on throughput performance in mobile ad hoc networks” ICC 2001. 2001 IEEE International Conference on Communications. Conference Record. Helsinky, Finland, June 11-14, 2001, IEEE International Conference on Communications, New York, NY: IEEE, US, vol. vol. 1 of 10, Jun. 11, 2001, pp. 2933-2937, XP 010553662 ISBN: 0-7803-7097-1.

Kim Y ED—Association for Computing Machinery: “Simple and Fault—Tolerant Key Agreement by Dynamic Collaborative Groups” Proceedings of the 7th ACM Conference on Computer and Communications Security. CS 2000. Athens, Greece, Nov. 1-4, 2000, ACM Conference on Computer and Communications Security, New Your, NY: ACM, US, vol. CONF. 7, Nov. 1, 2000, pp. 1-38, XP 002951317 ISBN: 1-58113-203-4.

Bolt, R.A., “Put-that-there”: voice and gesture at the graphics interface, Computer Graphics, vol. 14, No. 3 (ACM SIGGRAPH Conference Proceedings) Jul. 1980, pp. 262-270.

DeWitt, Thomas and Edelstein, Phil, “Pantomation: A System for Position Tracking,” Proceedings of the 2nd Symposium on Small Computers in the Arts, Oct. 1982, pp. 61-69.

* cited by examiner

FIG. 1

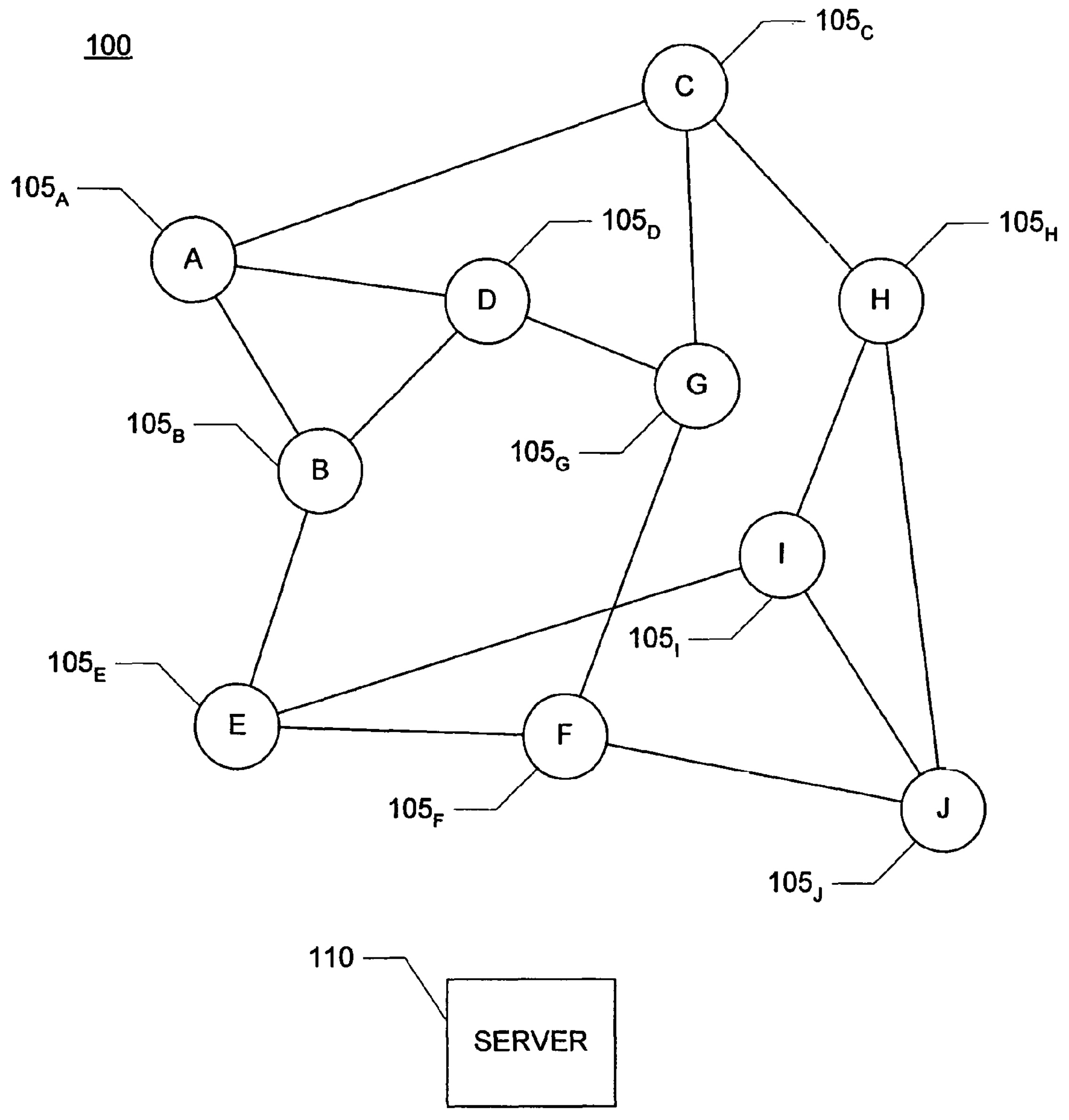

100
105_C
C
105_A
A
105_D
D
105_H
H
G
105_B
B
105_G
I
105_I
105_E
E
F
J
105_F
105_J
110
SERVER

FIG. 2

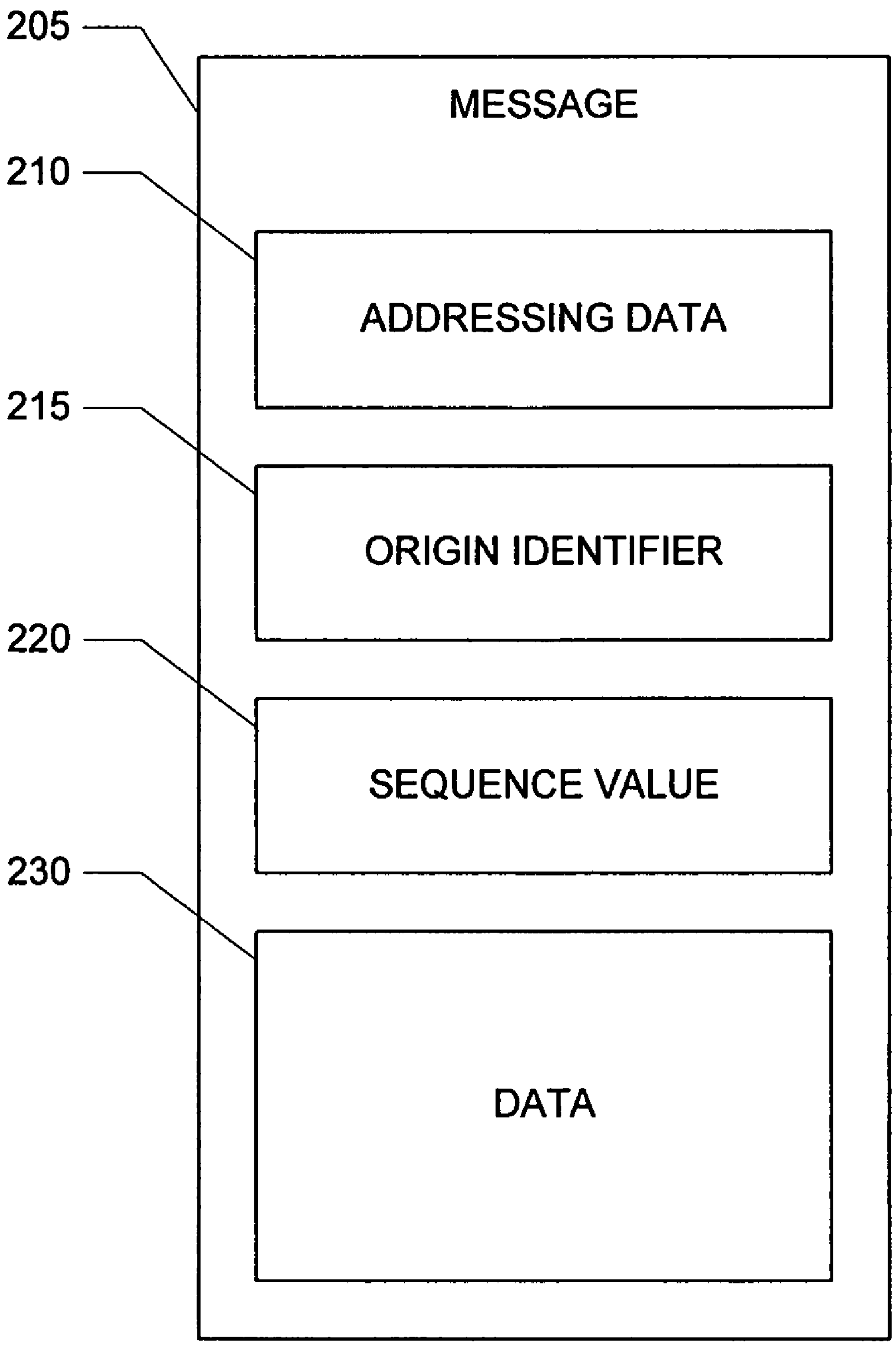

205
MESSAGE
210
ADDRESSING DATA
215
ORIGIN IDENTIFIER
220
SEQUENCE VALUE
230
DATA

FIG. 3

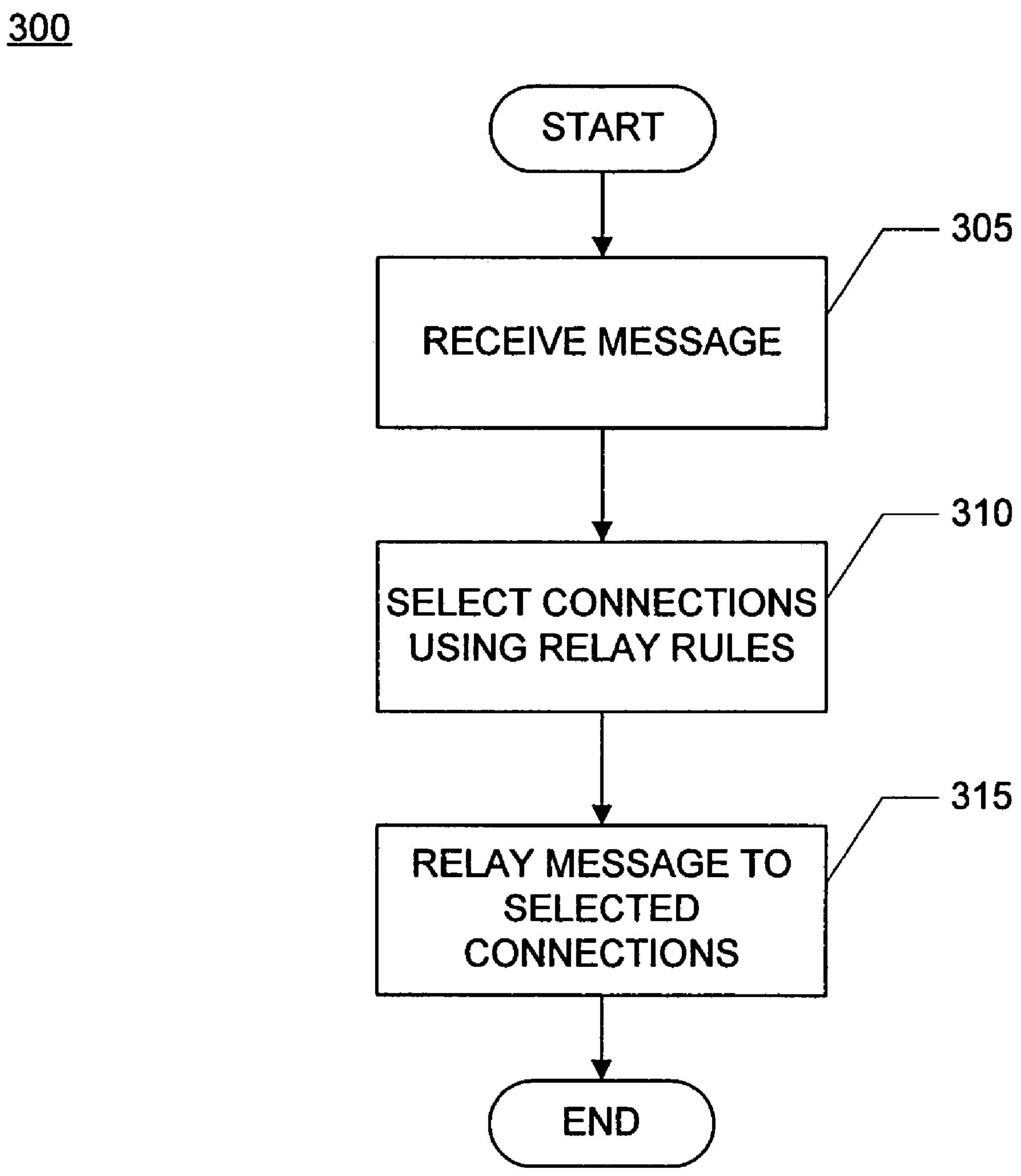

300
START
305
RECEIVE MESSAGE
310
SELECT CONNECTIONS USING RELAY RULES
315
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 4

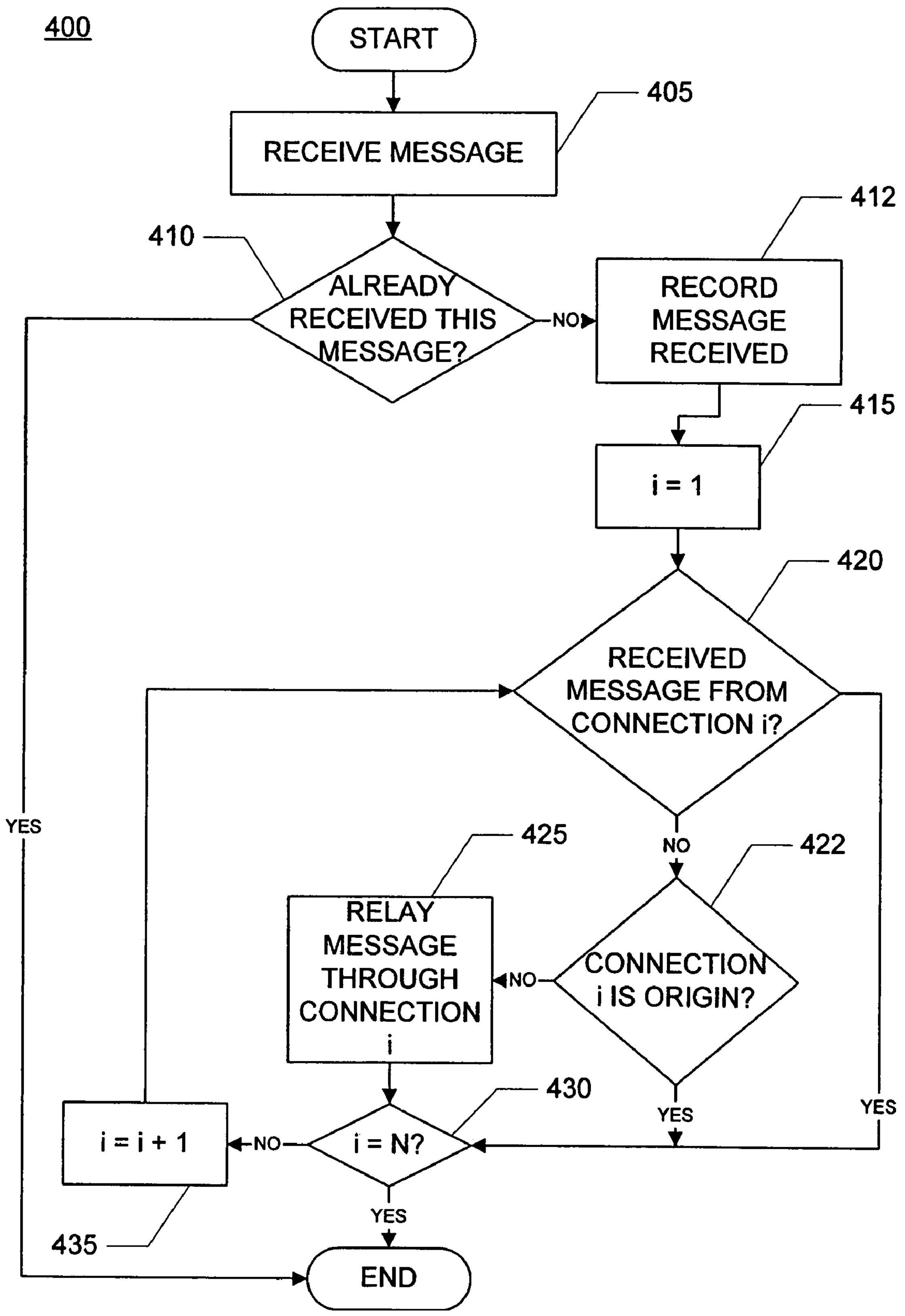

400
START
RECEIVE MESSAGE
405
410
ALREADY RECEIVED THIS MESSAGE?
NO
412
RECORD MESSAGE RECEIVED
415
i = 1
420
RECEIVED MESSAGE FROM CONNECTION i?
YES
NO
425
422
RELAY MESSAGE THROUGH CONNECTION i
NO
CONNECTION i IS ORIGIN?
YES
430
i = i + 1
NO
i = N?
YES
435
END

FIG. 5

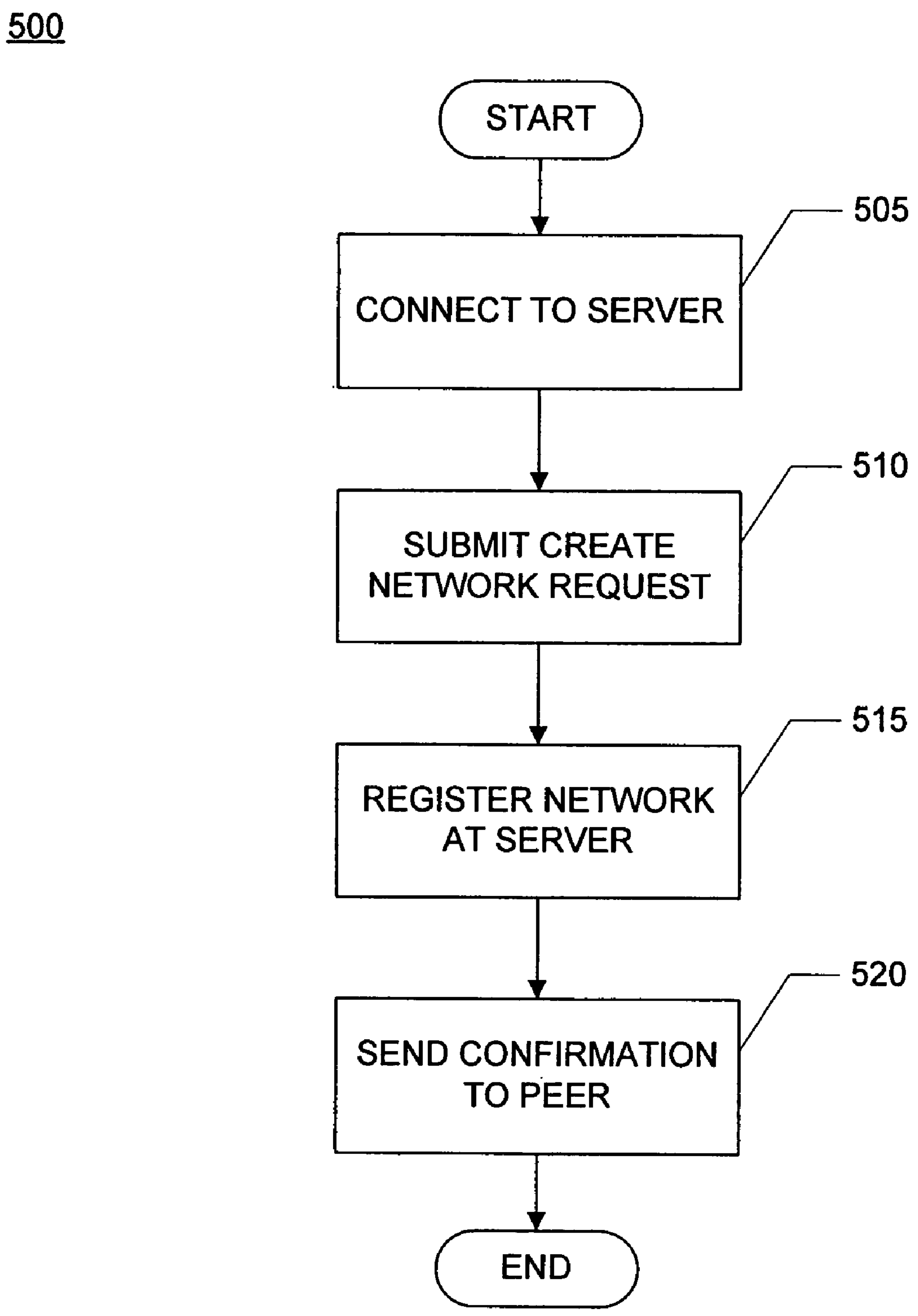

500
START
505
CONNECT TO SERVER
510
SUBMIT CREATE NETWORK REQUEST
515
REGISTER NETWORK AT SERVER
520
SEND CONFIRMATION TO PEER
END

FIG. 6

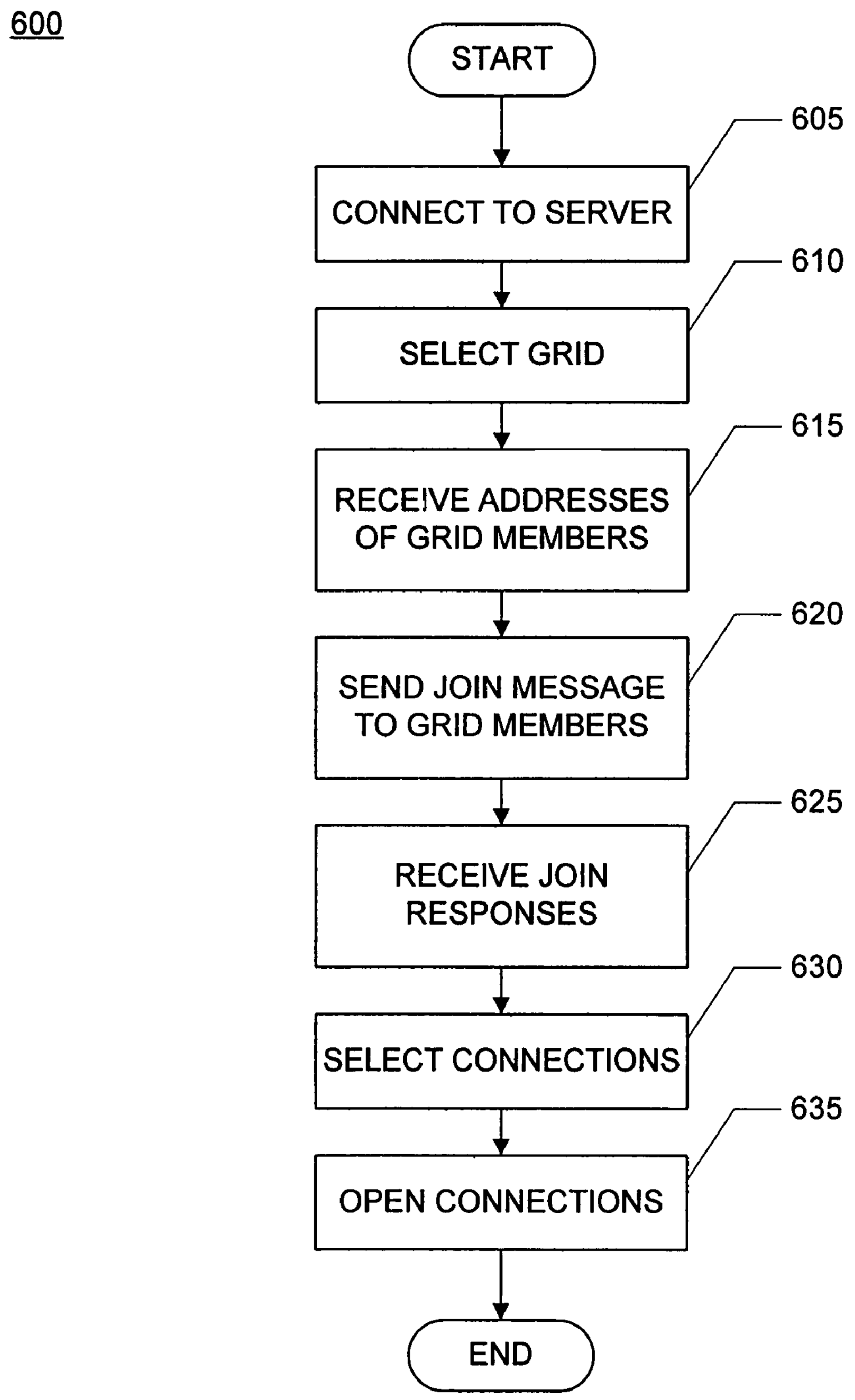

600
START
CONNECT TO SERVER
605
SELECT GRID
610
RECEIVE ADDRESSES OF GRID MEMBERS
615
SEND JOIN MESSAGE TO GRID MEMBERS
620
RECEIVE JOIN RESPONSES
625
SELECT CONNECTIONS
630
OPEN CONNECTIONS
635
END

FIG. 7

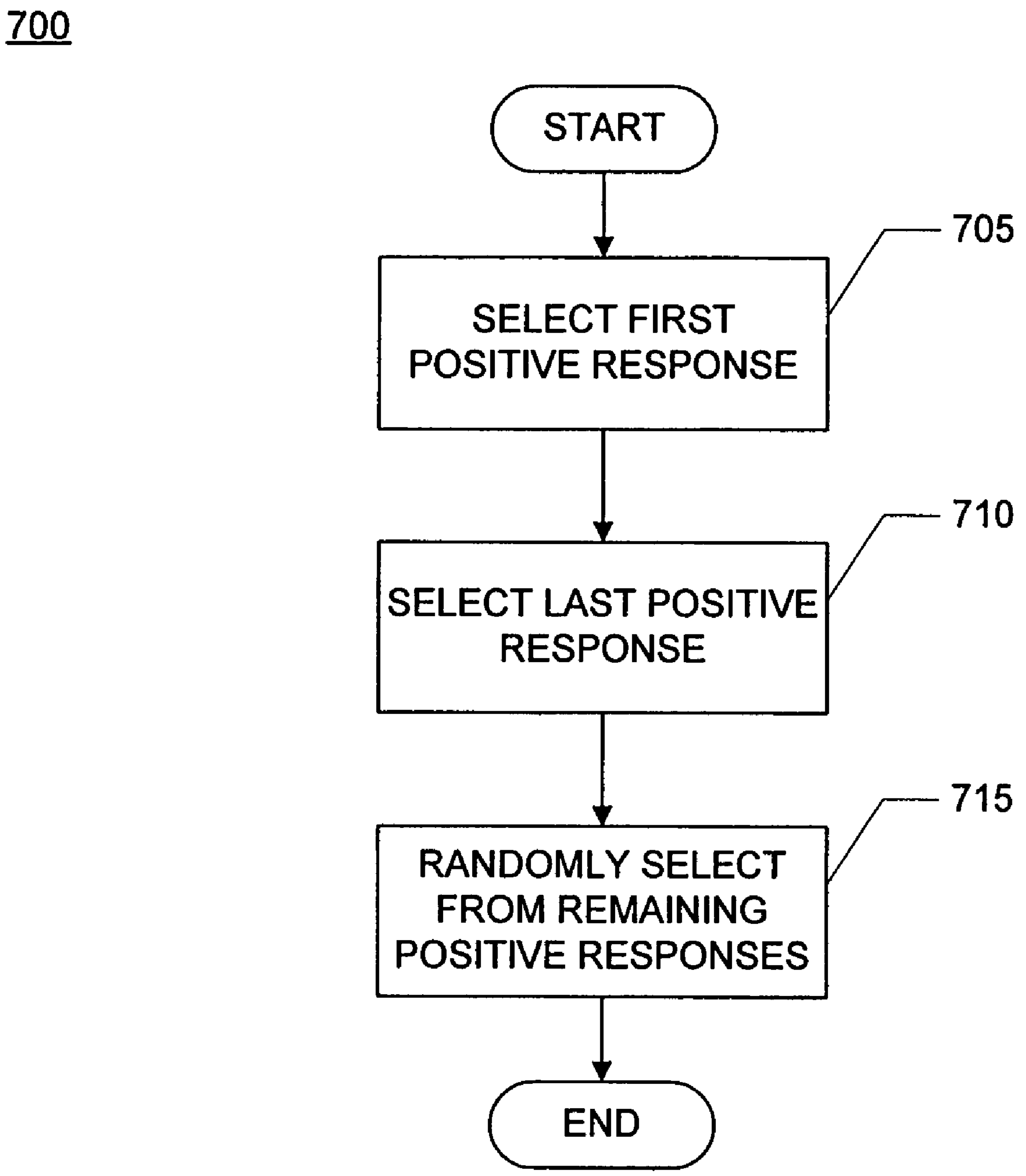

700
START
SELECT FIRST POSITIVE RESPONSE
705
SELECT LAST POSITIVE RESPONSE
710
RANDOMLY SELECT FROM REMAINING POSITIVE RESPONSES
715
END

FIG. 8

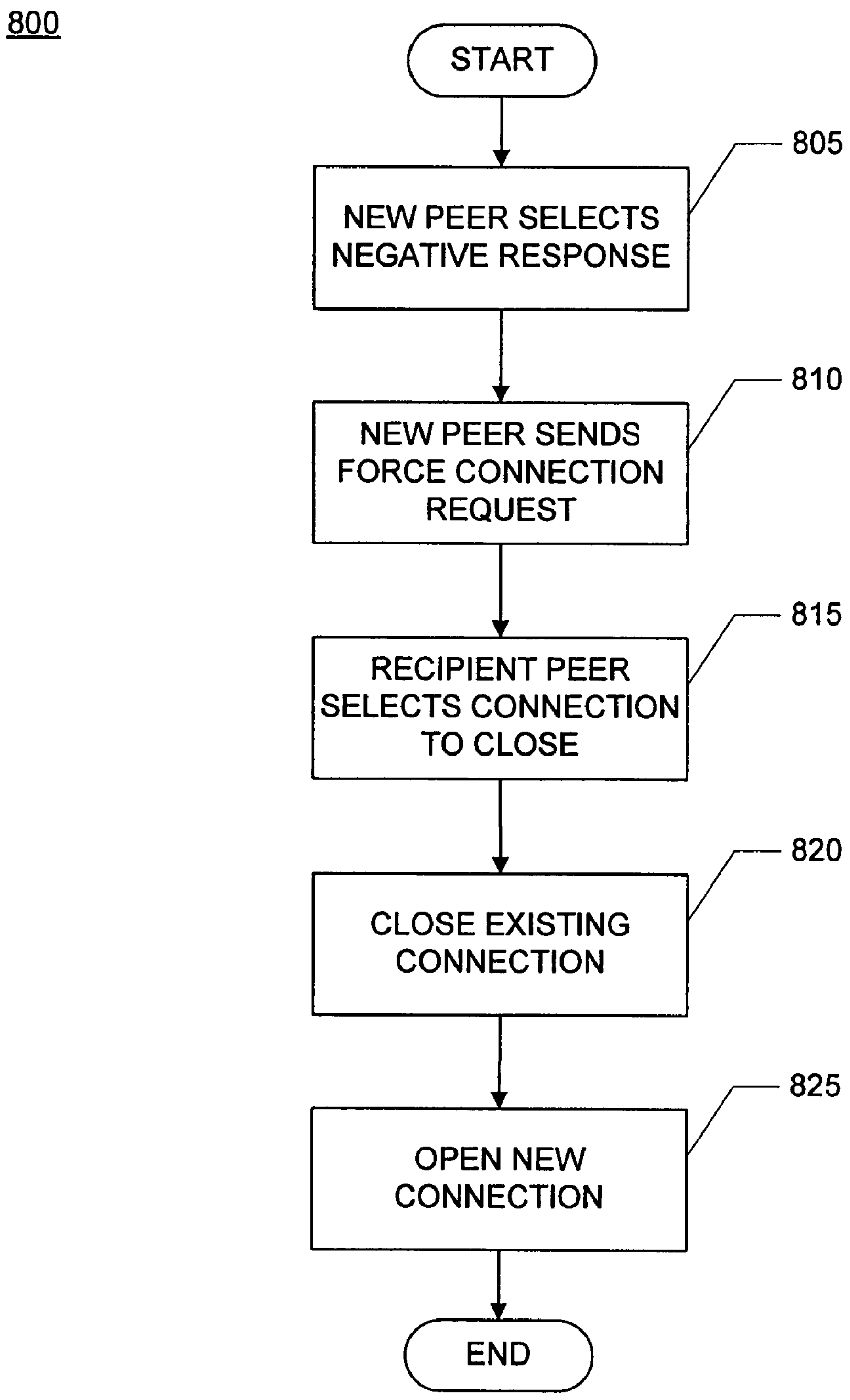

800
START
805
NEW PEER SELECTS NEGATIVE RESPONSE
810
NEW PEER SENDS FORCE CONNECTION REQUEST
815
RECIPIENT PEER SELECTS CONNECTION TO CLOSE
820
CLOSE EXISTING CONNECTION
825
OPEN NEW CONNECTION
END

FIG. 9

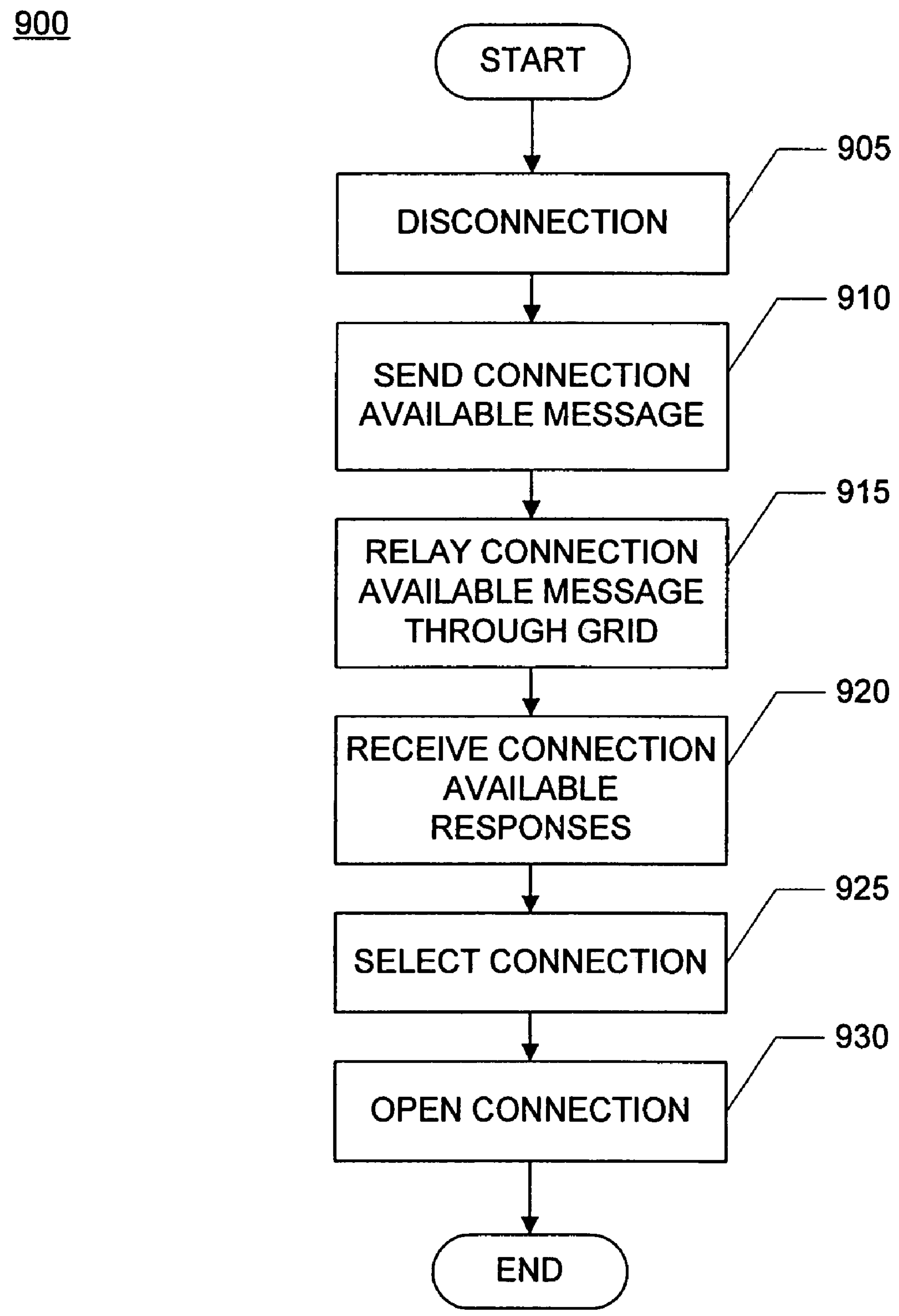

900
START
DISCONNECTION
905
SEND CONNECTION AVAILABLE MESSAGE
910
RELAY CONNECTION AVAILABLE MESSAGE THROUGH GRID
915
RECEIVE CONNECTION AVAILABLE RESPONSES
920
SELECT CONNECTION
925
OPEN CONNECTION
930
END

FIG. 10

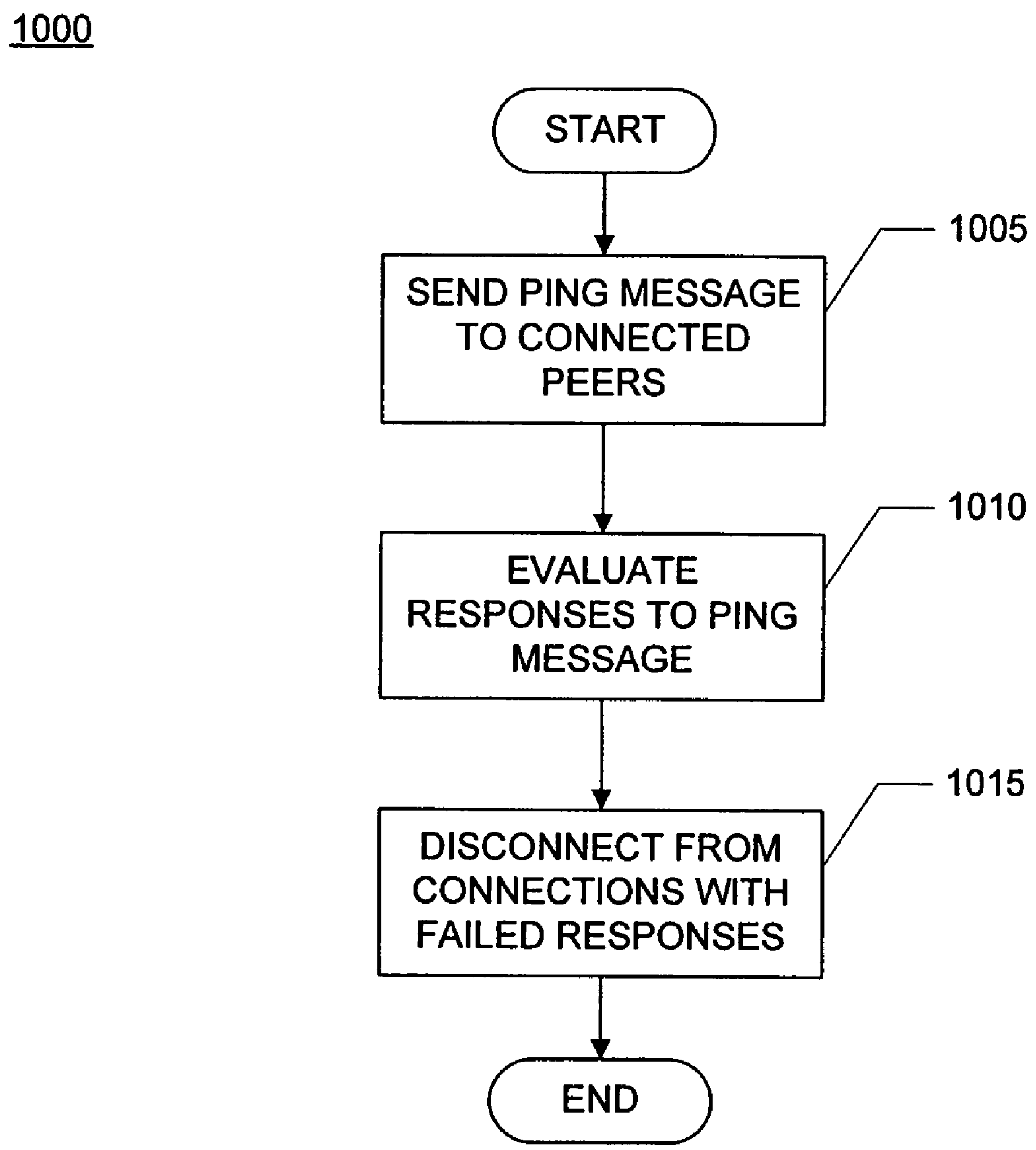

1000
START
SEND PING MESSAGE TO CONNECTED PEERS
1005
EVALUATE RESPONSES TO PING MESSAGE
1010
DISCONNECT FROM CONNECTIONS WITH FAILED RESPONSES
1015
END

FIG. 11

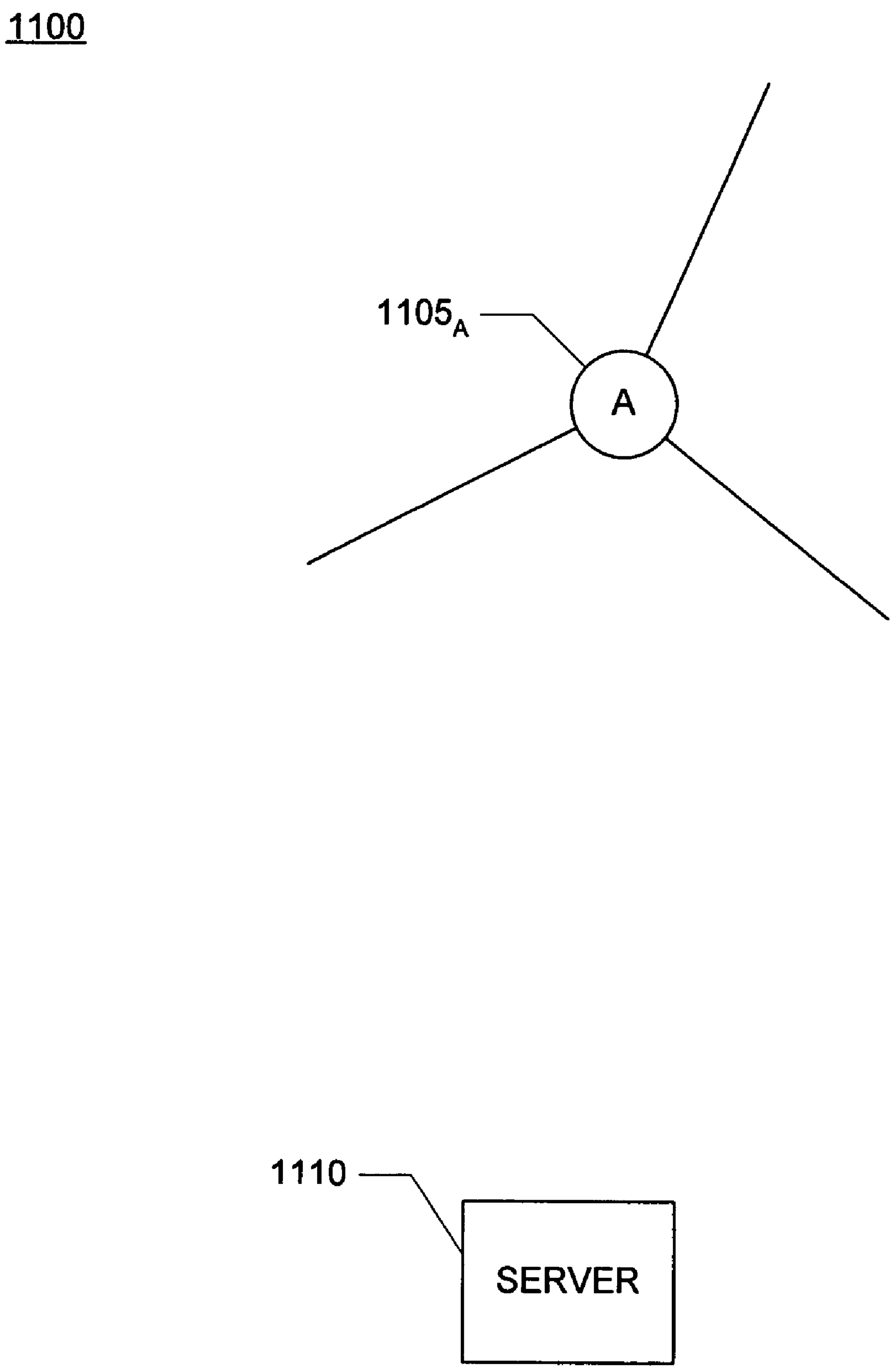


FIG. 12

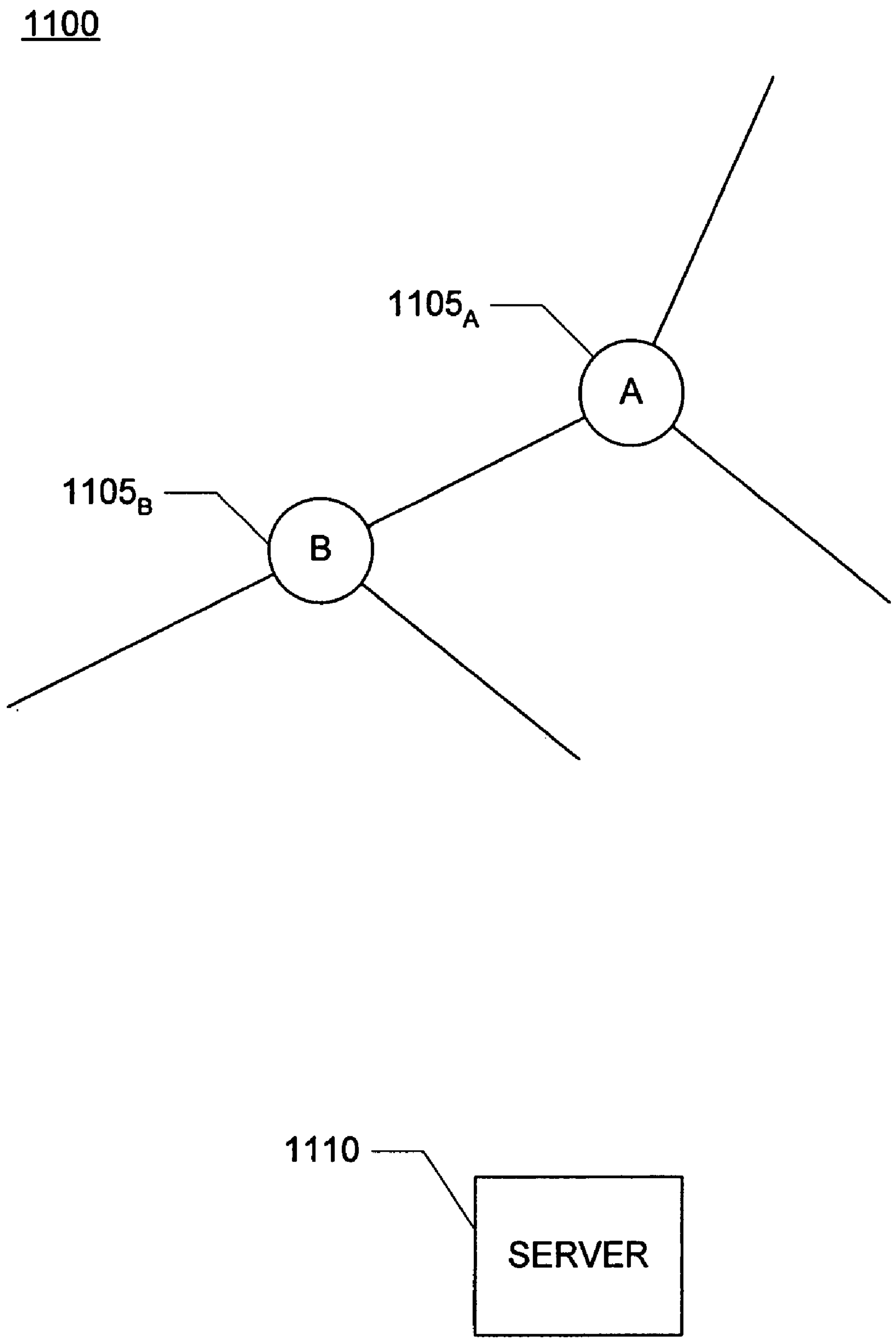


FIG. 13

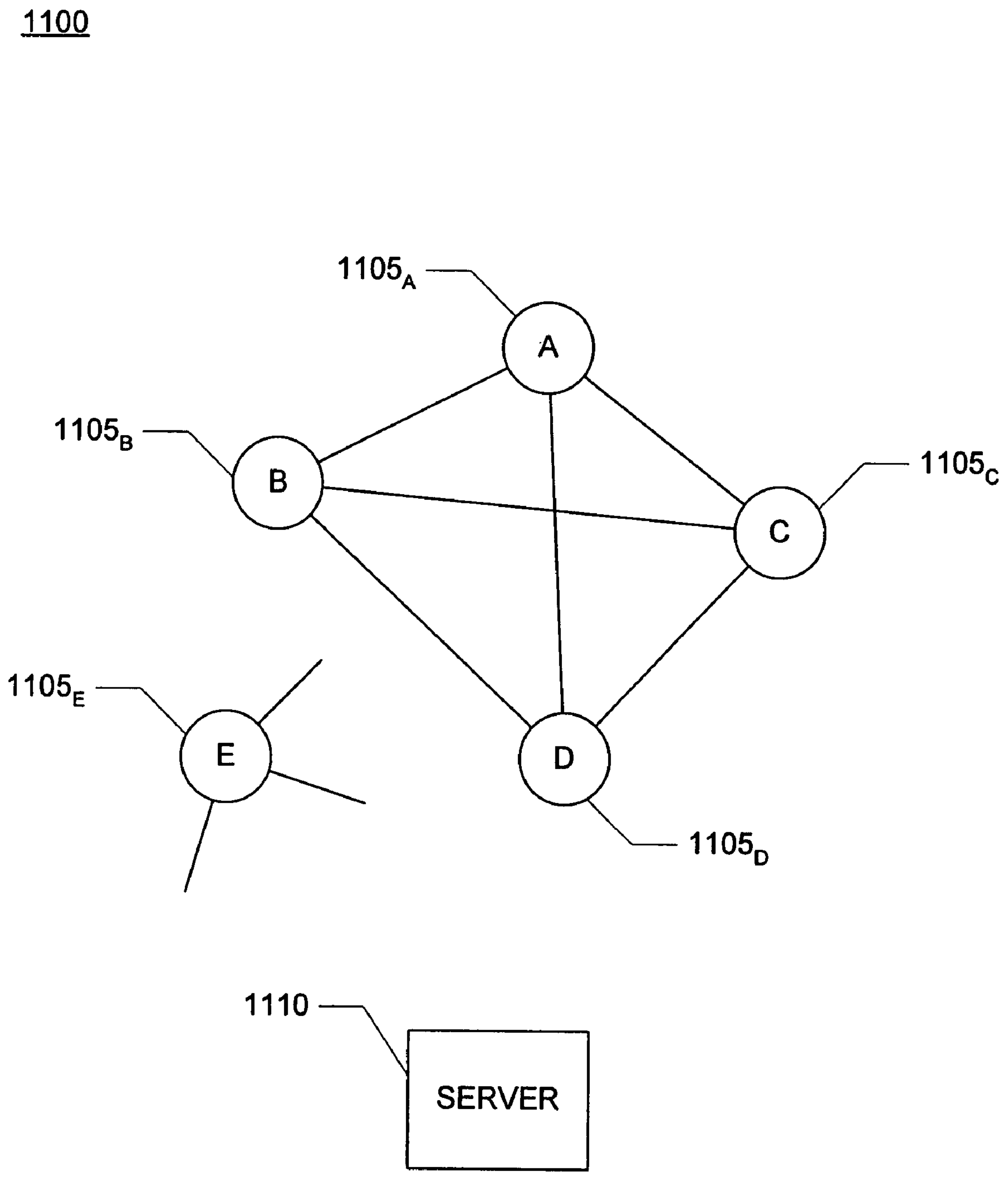


FIG. 14

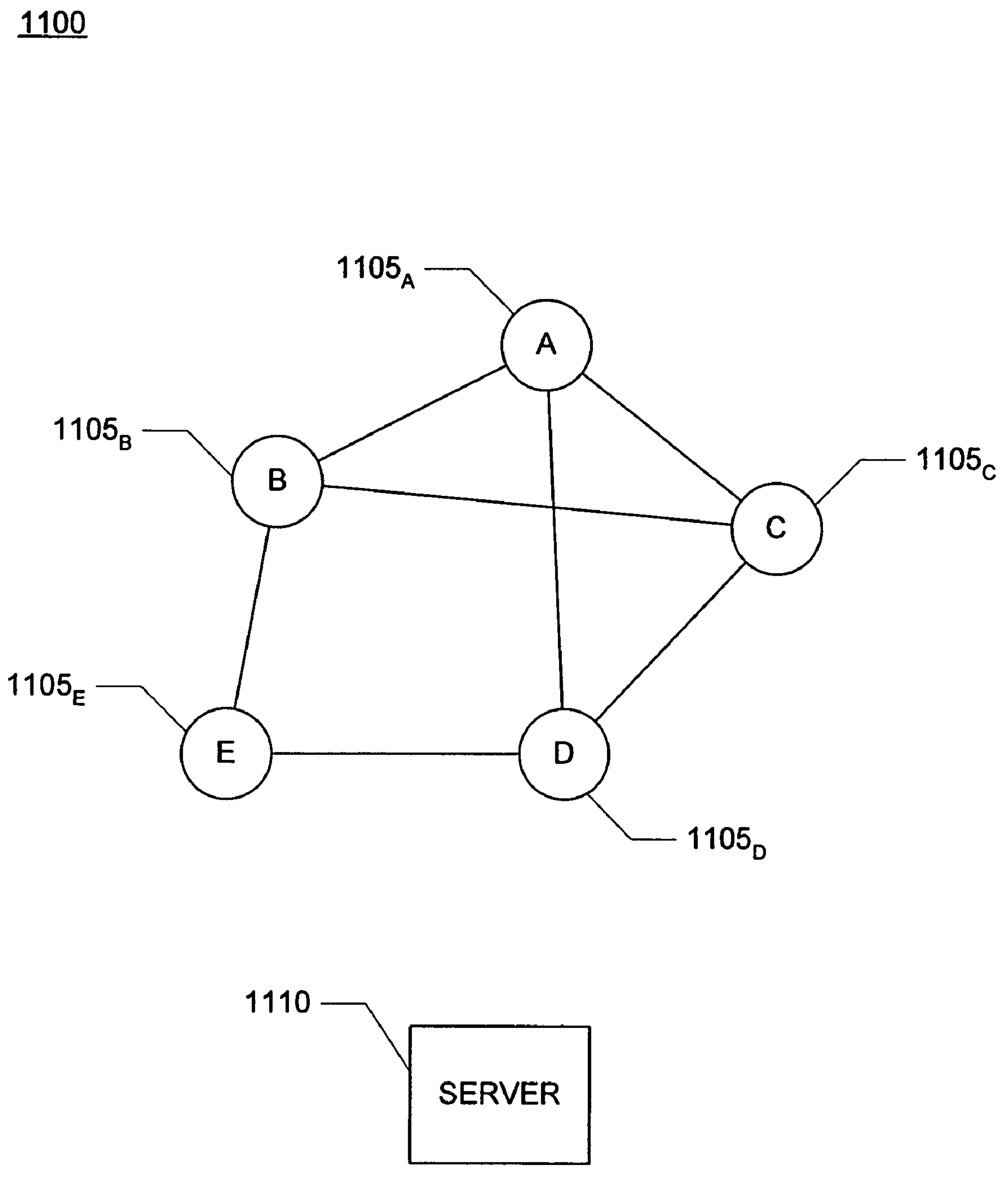


FIG. 15

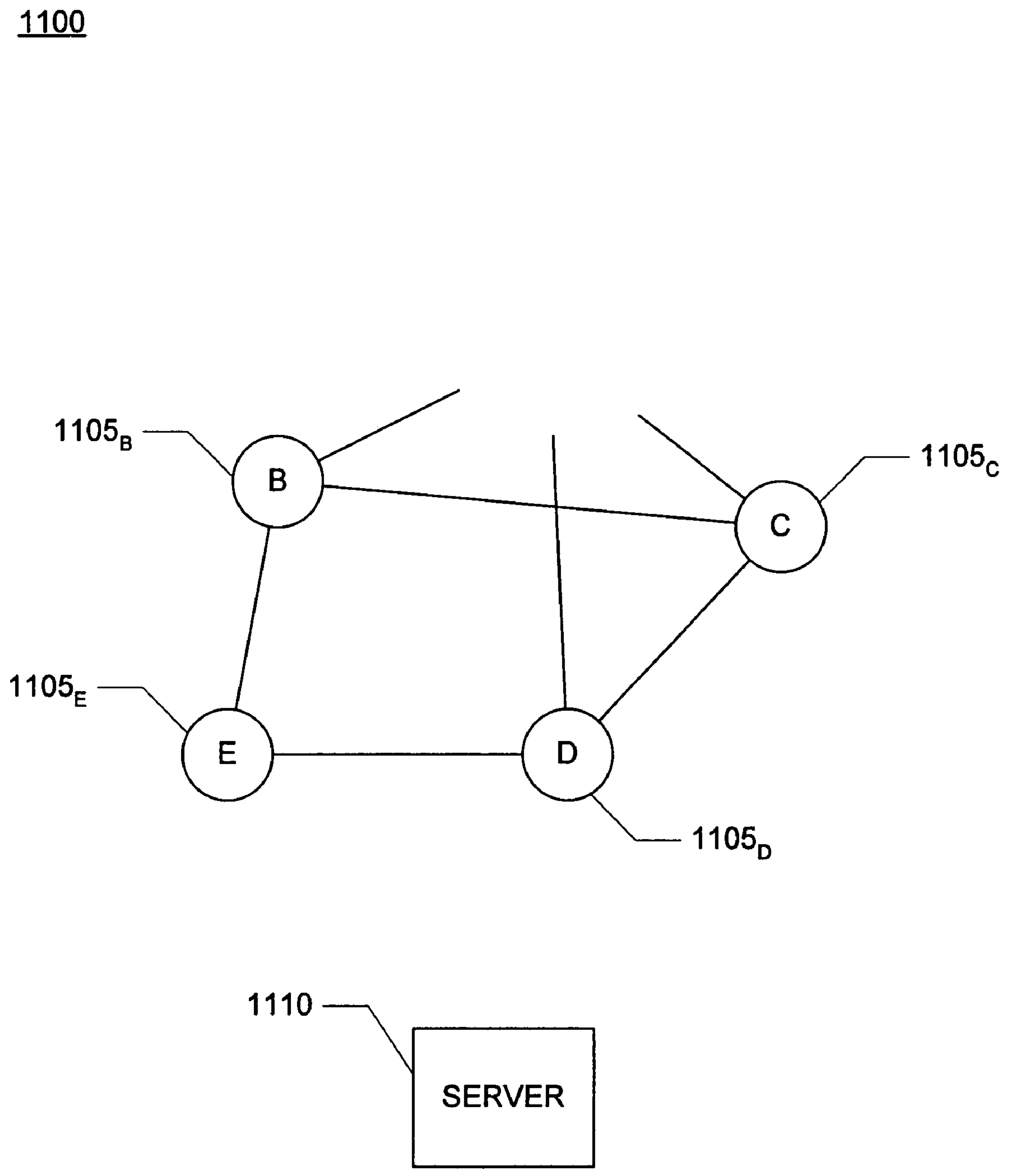


FIG. 16

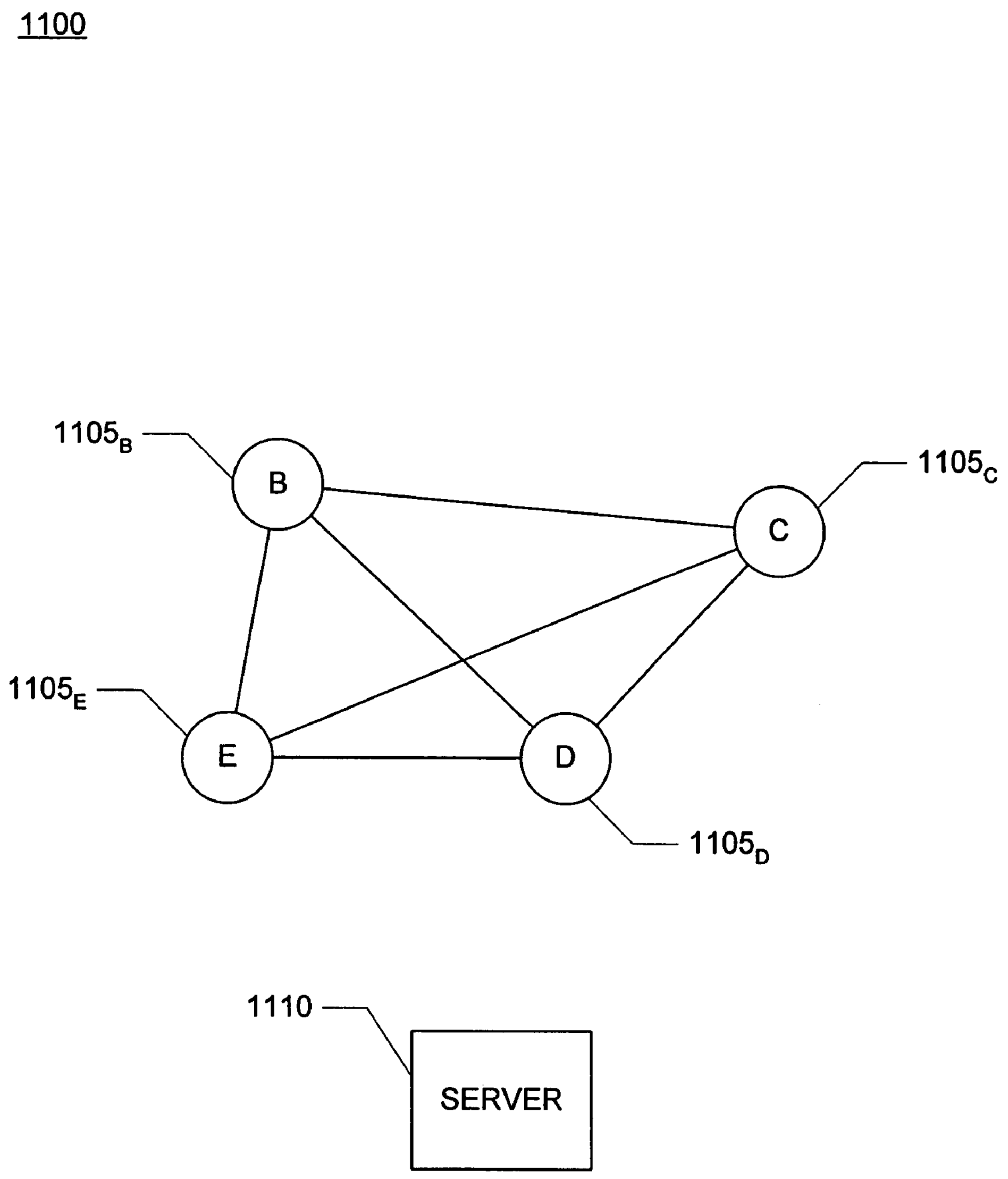


FIG. 17

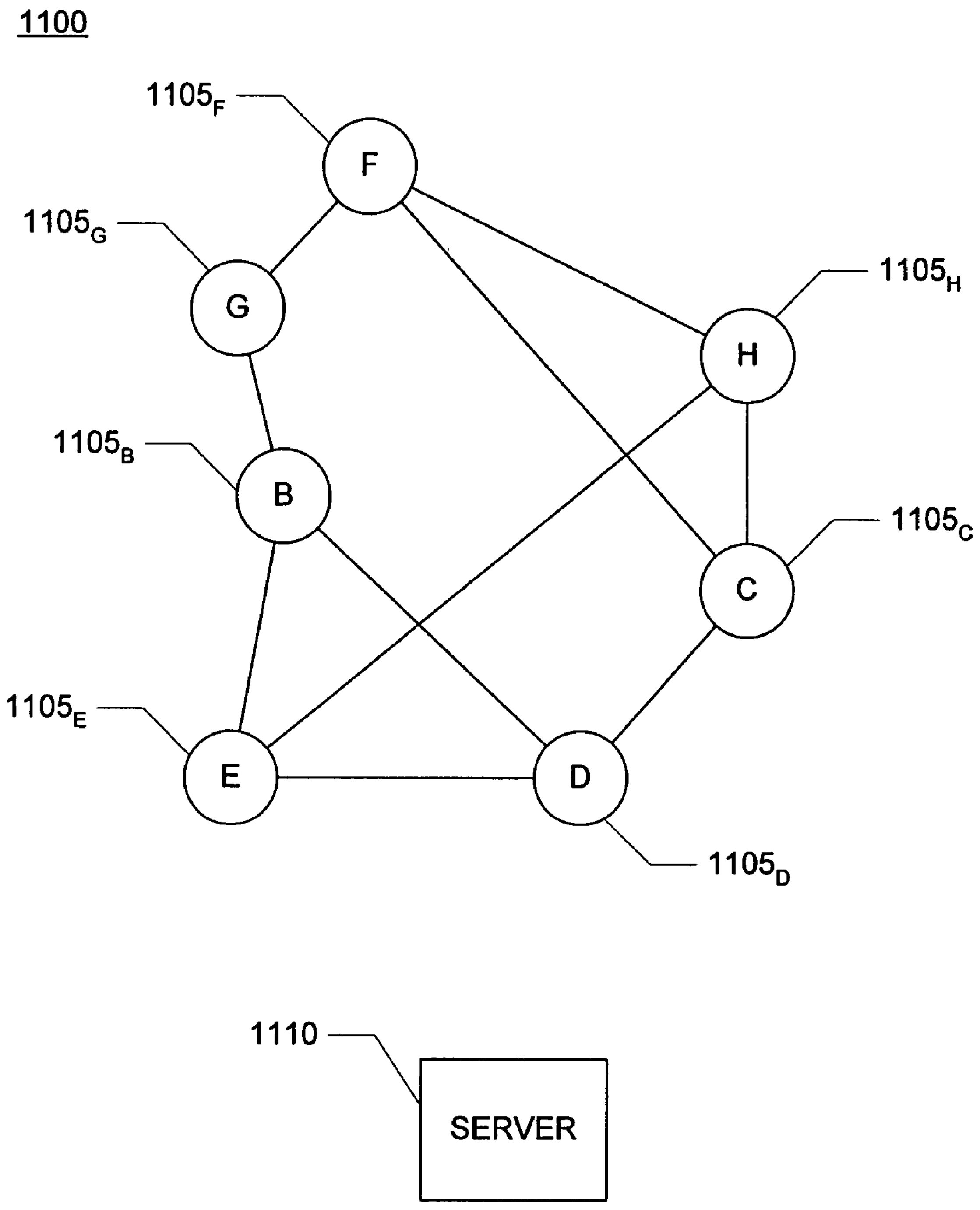

1100
1105_F
F
1105_G
G
1105_H
H
1105_B
B
1105_C
C
1105_E
E
D
1105_D
1110
SERVER

FIG. 18

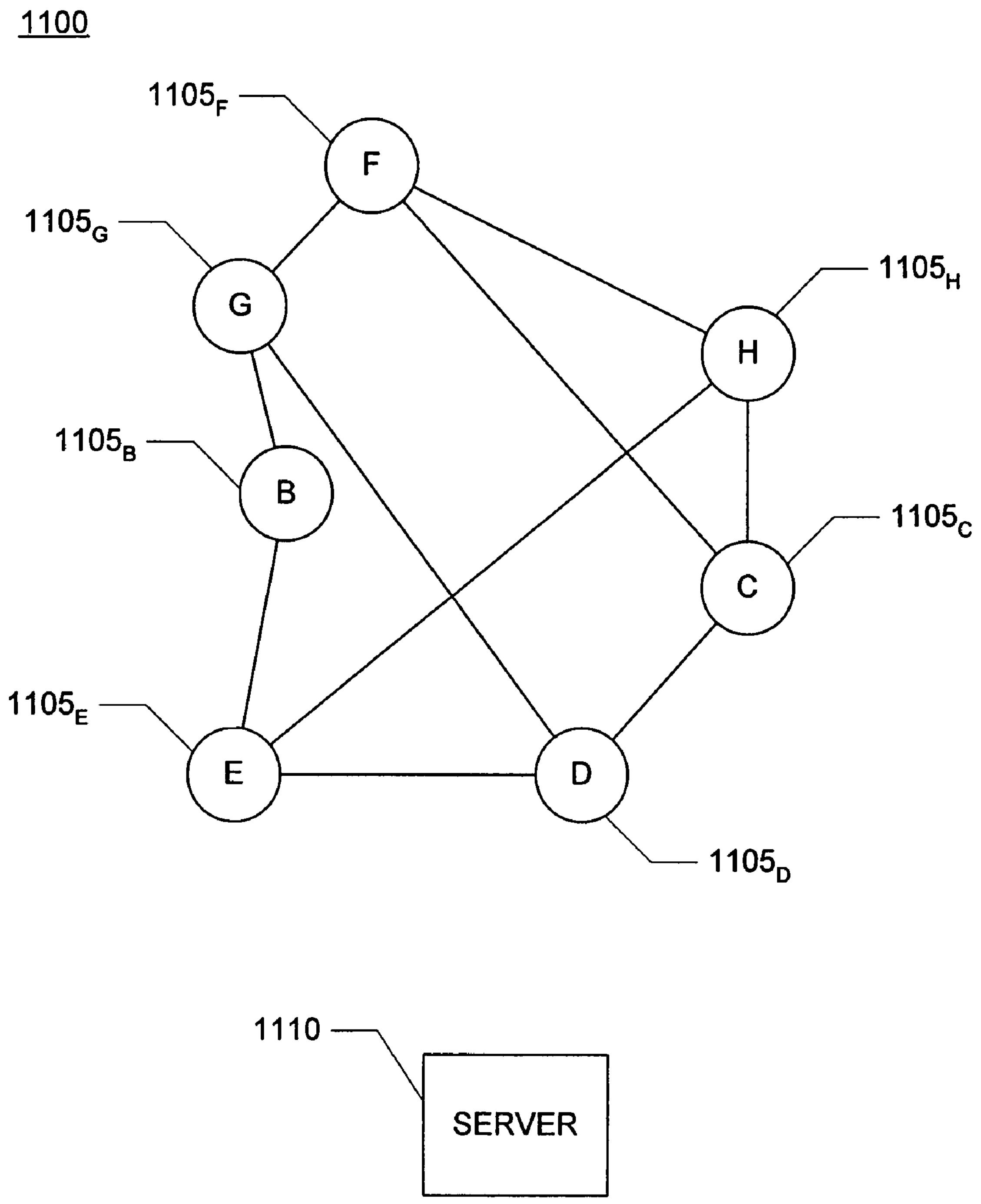

1100
1105F
F
1105G
G
1105H
H
1105B
B
1105C
C
1105E
E
D
1105D
1110
SERVER

FIG. 19

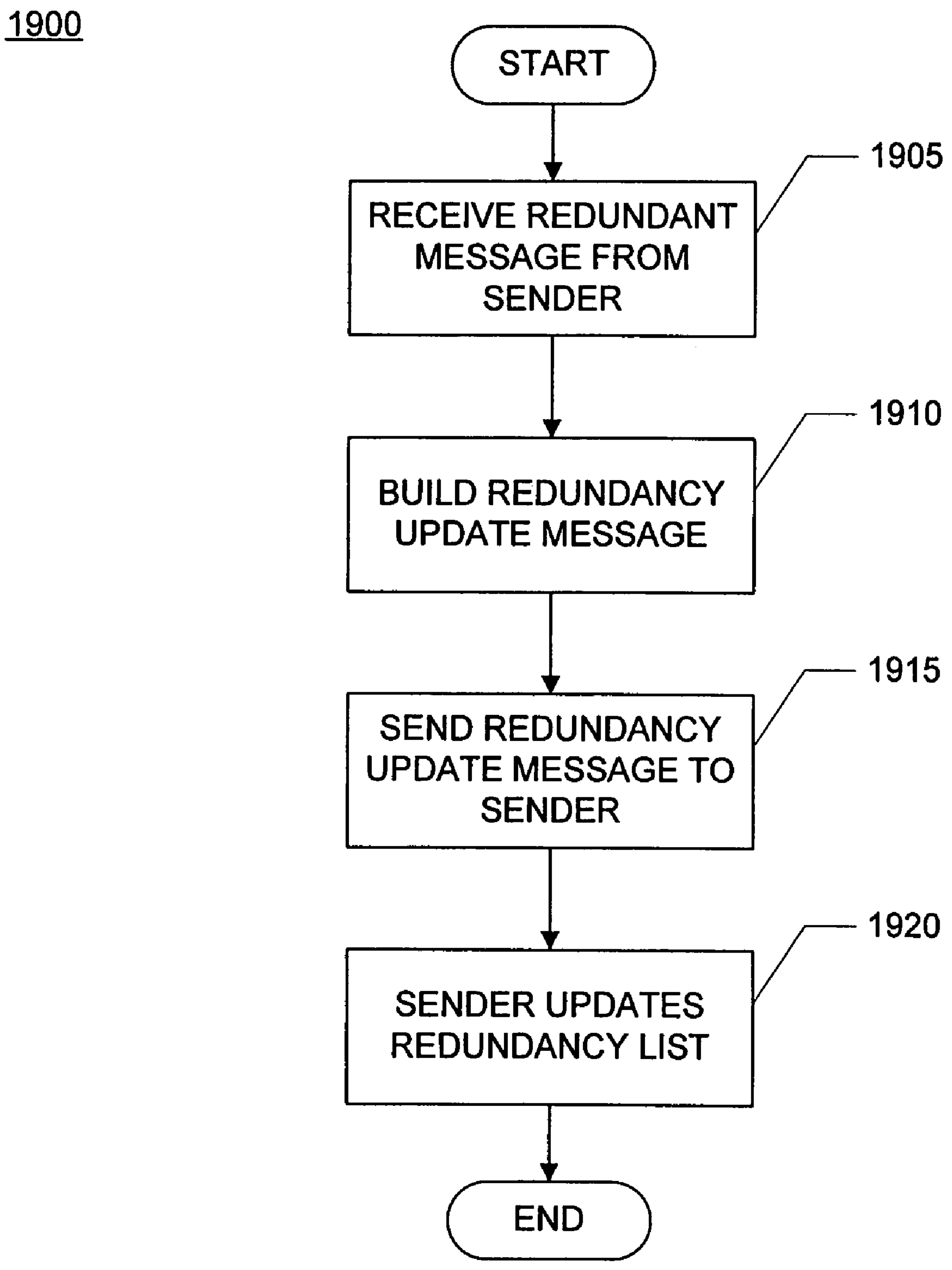

1900
START
RECEIVE REDUNDANT MESSAGE FROM SENDER
1905
BUILD REDUNDANCY UPDATE MESSAGE
1910
SEND REDUNDANCY UPDATE MESSAGE TO SENDER
1915
SENDER UPDATES REDUNDANCY LIST
1920
END

FIG. 20

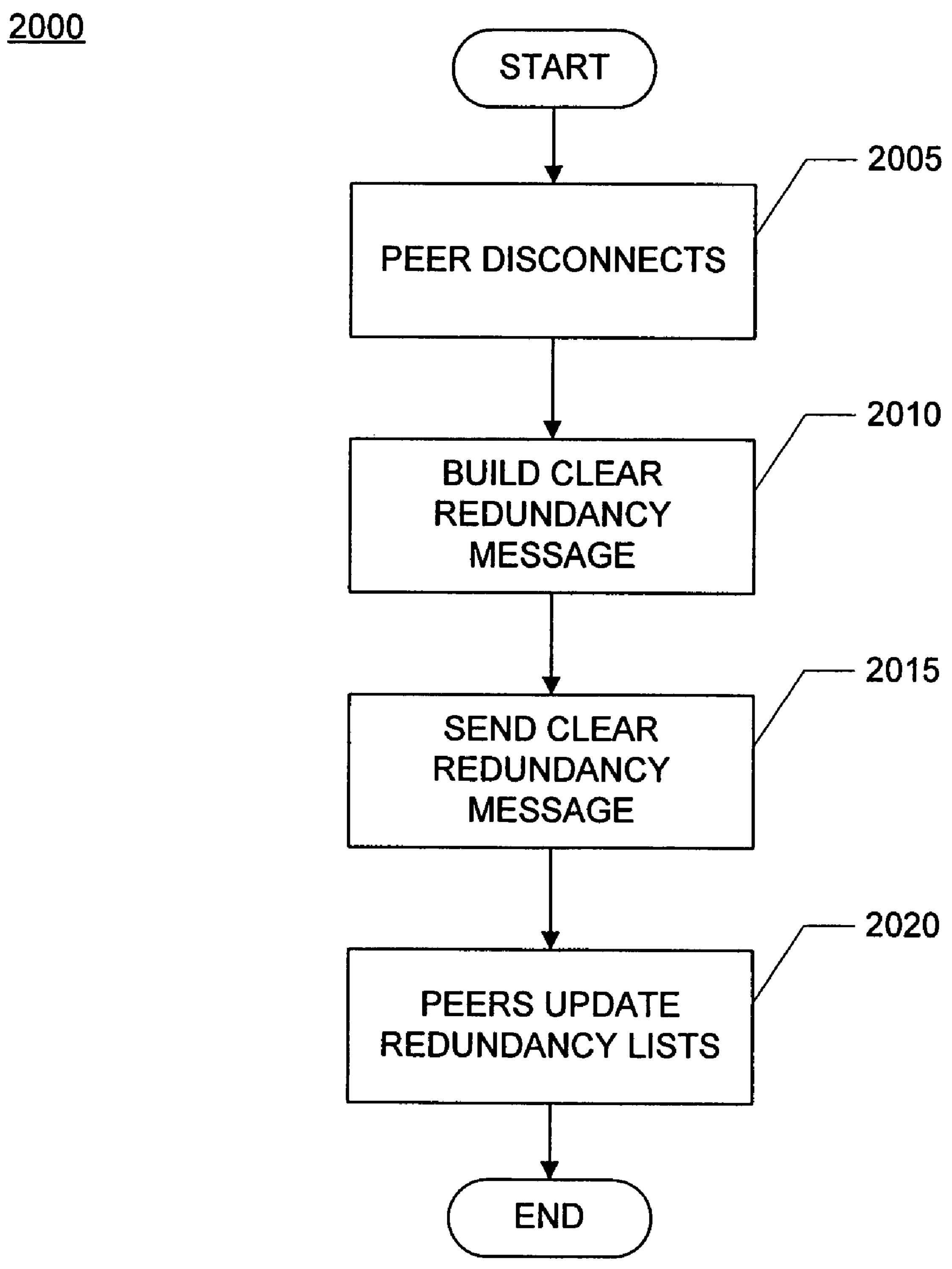

2000
START
2005
PEER DISCONNECTS
2010
BUILD CLEAR REDUNDANCY MESSAGE
2015
SEND CLEAR REDUNDANCY MESSAGE
2020
PEERS UPDATE REDUNDANCY LISTS
END

FIG. 21

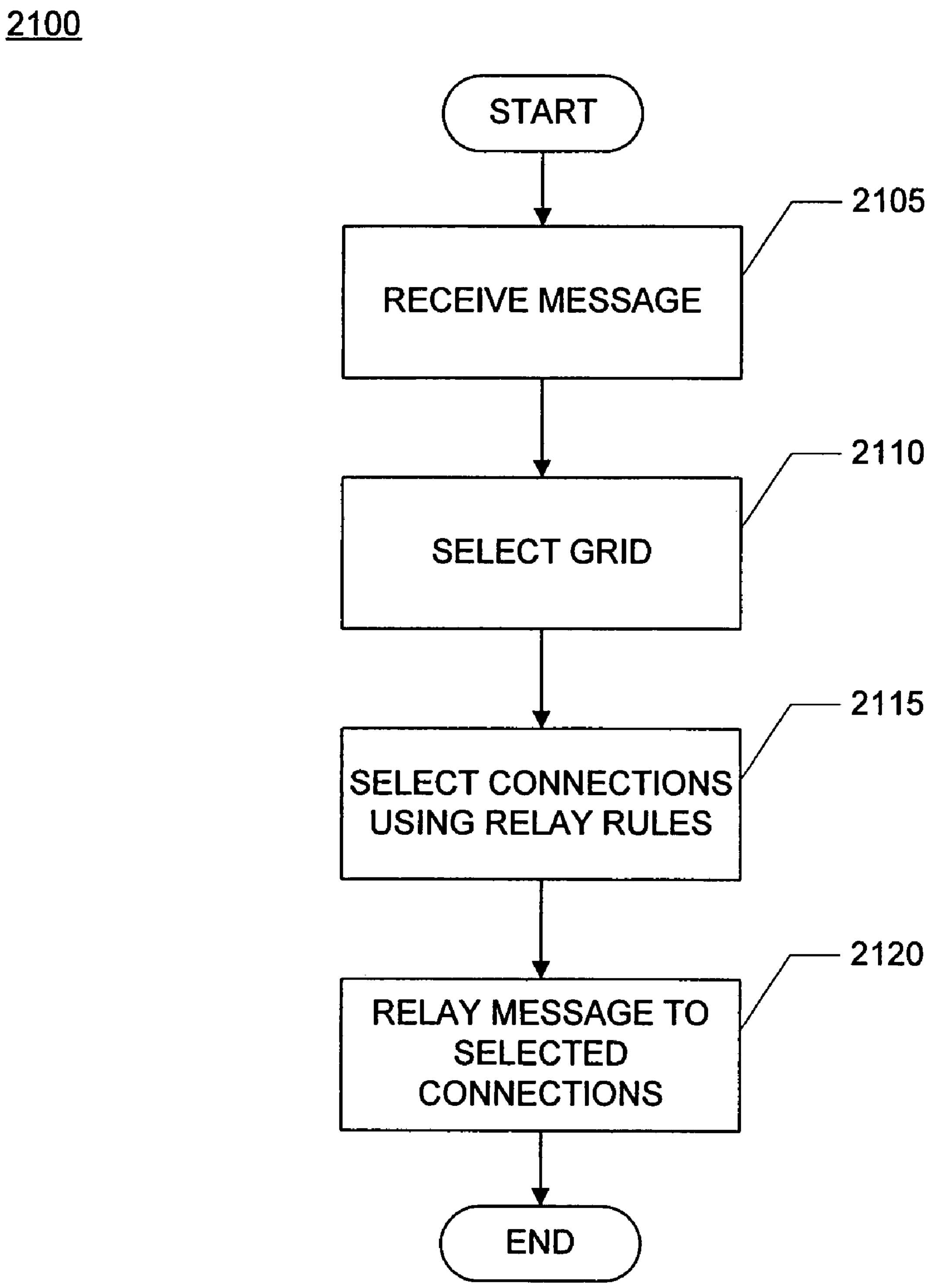

2100
START
RECEIVE MESSAGE
2105
SELECT GRID
2110
SELECT CONNECTIONS USING RELAY RULES
2115
RELAY MESSAGE TO SELECTED CONNECTIONS
2120
END

FIG. 22

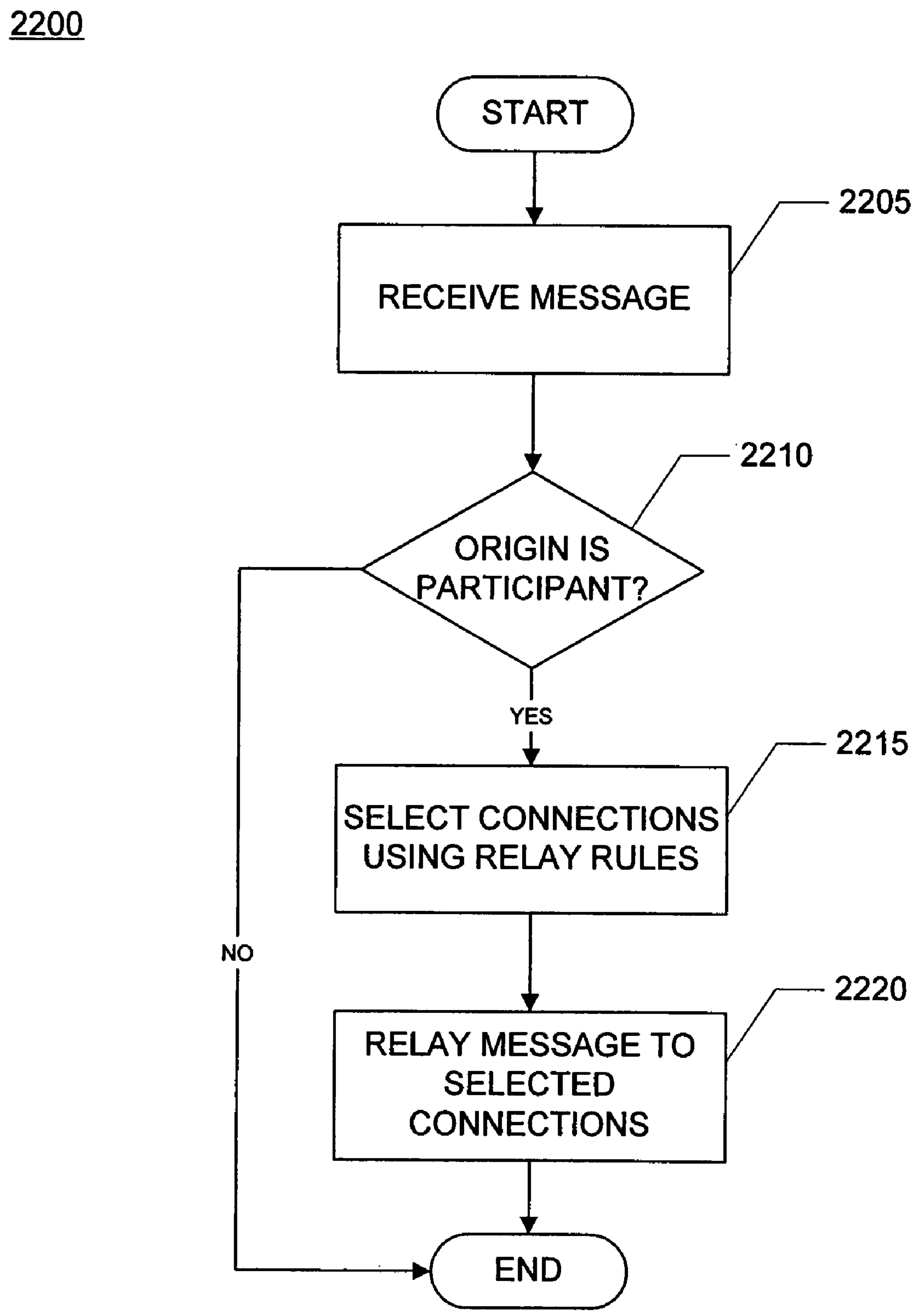

2200
START
2205
RECEIVE MESSAGE
2210
ORIGIN IS PARTICIPANT?
YES
NO
2215
SELECT CONNECTIONS USING RELAY RULES
2220
RELAY MESSAGE TO SELECTED CONNECTIONS
END

FIG. 23

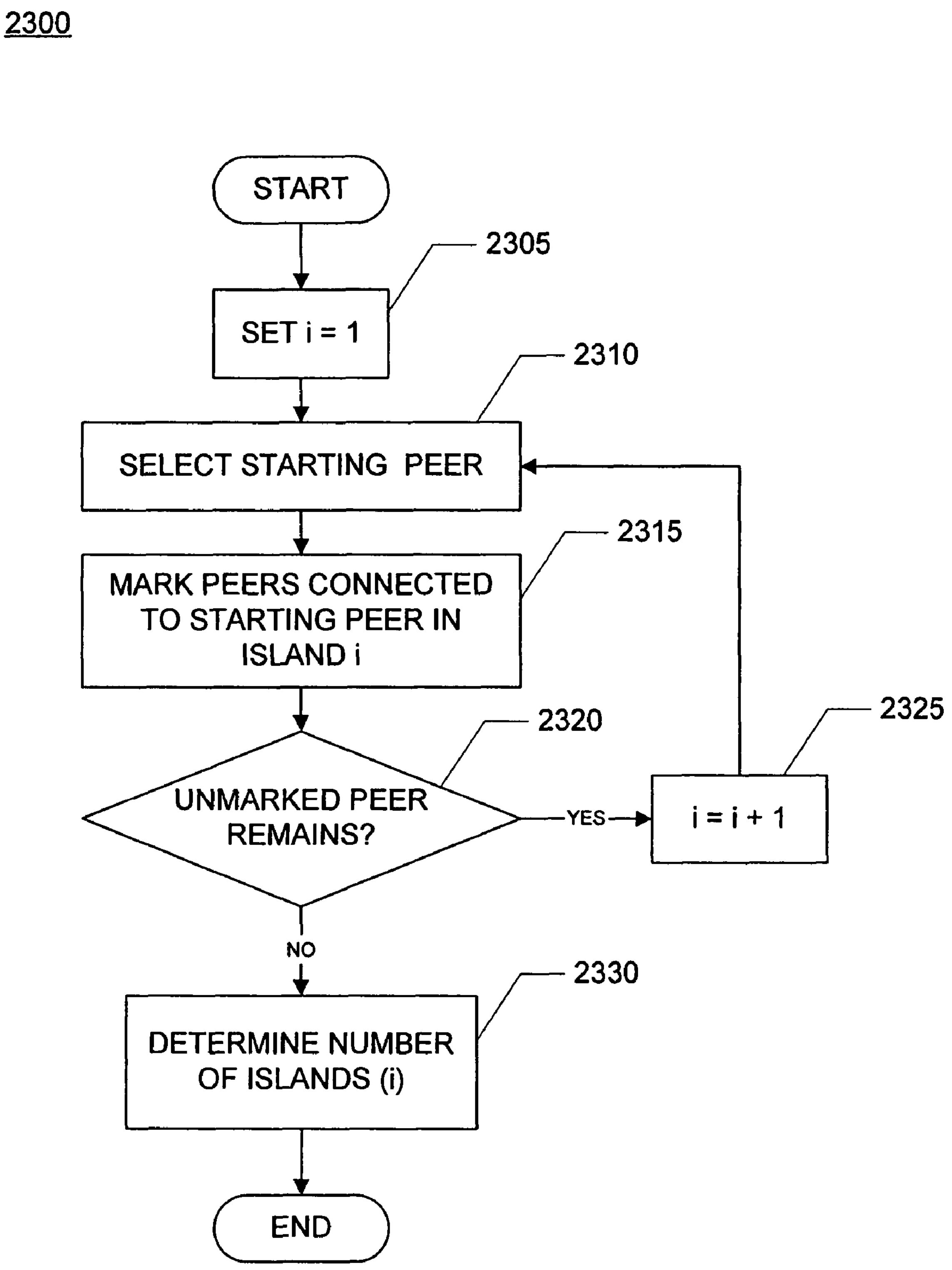

2300
START
2305
SET i = 1
2310
SELECT STARTING PEER
2315
MARK PEERS CONNECTED TO STARTING PEER IN ISLAND i
2320
UNMARKED PEER REMAINS?
YES
2325
i = i + 1
NO
2330
DETERMINE NUMBER OF ISLANDS (i)
END

FIG. 24

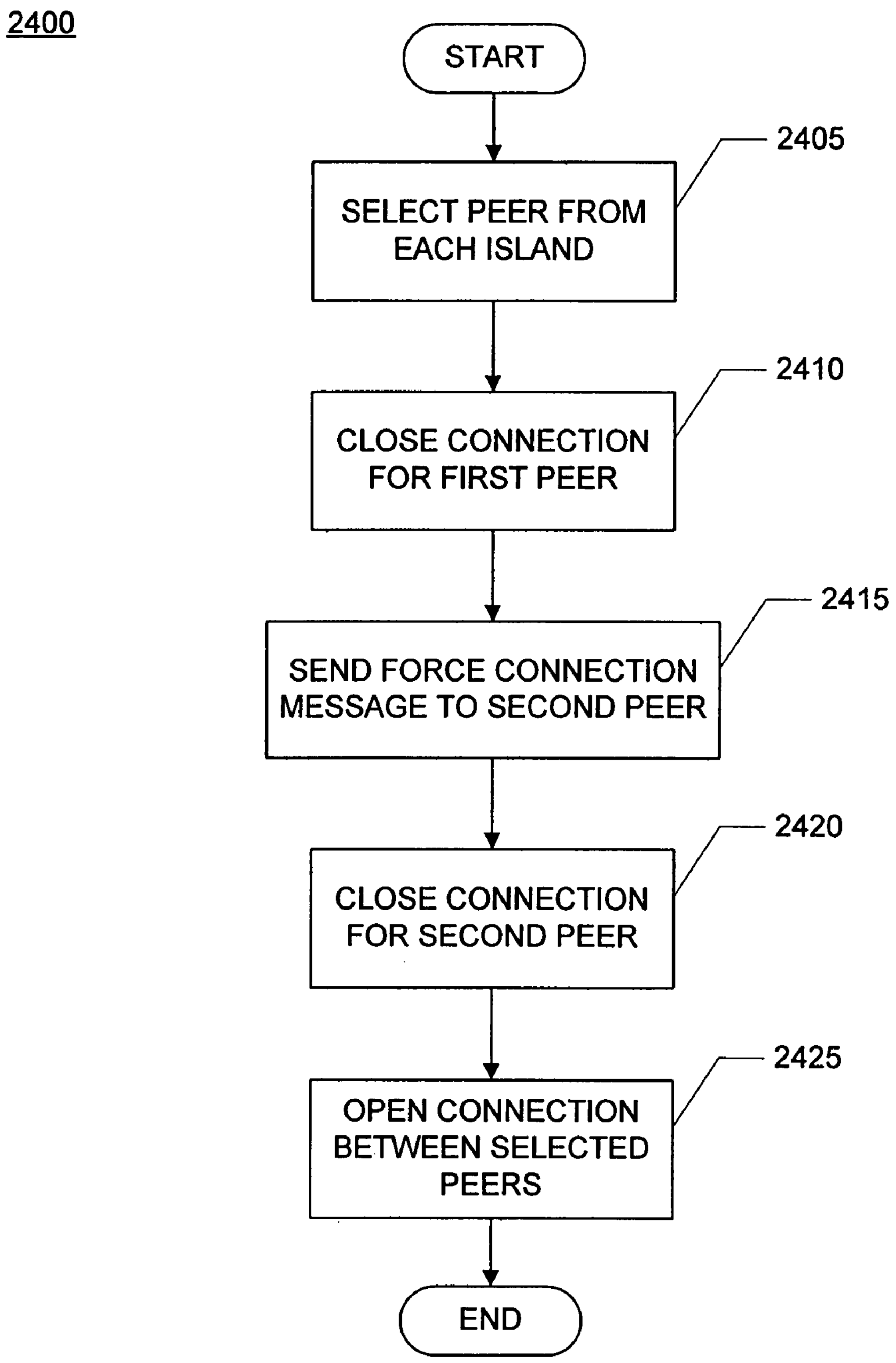

2400
START
2405
SELECT PEER FROM EACH ISLAND
2410
CLOSE CONNECTION FOR FIRST PEER
2415
SEND FORCE CONNECTION MESSAGE TO SECOND PEER
2420
CLOSE CONNECTION FOR SECOND PEER
2425
OPEN CONNECTION BETWEEN SELECTED PEERS
END

FIG. 25

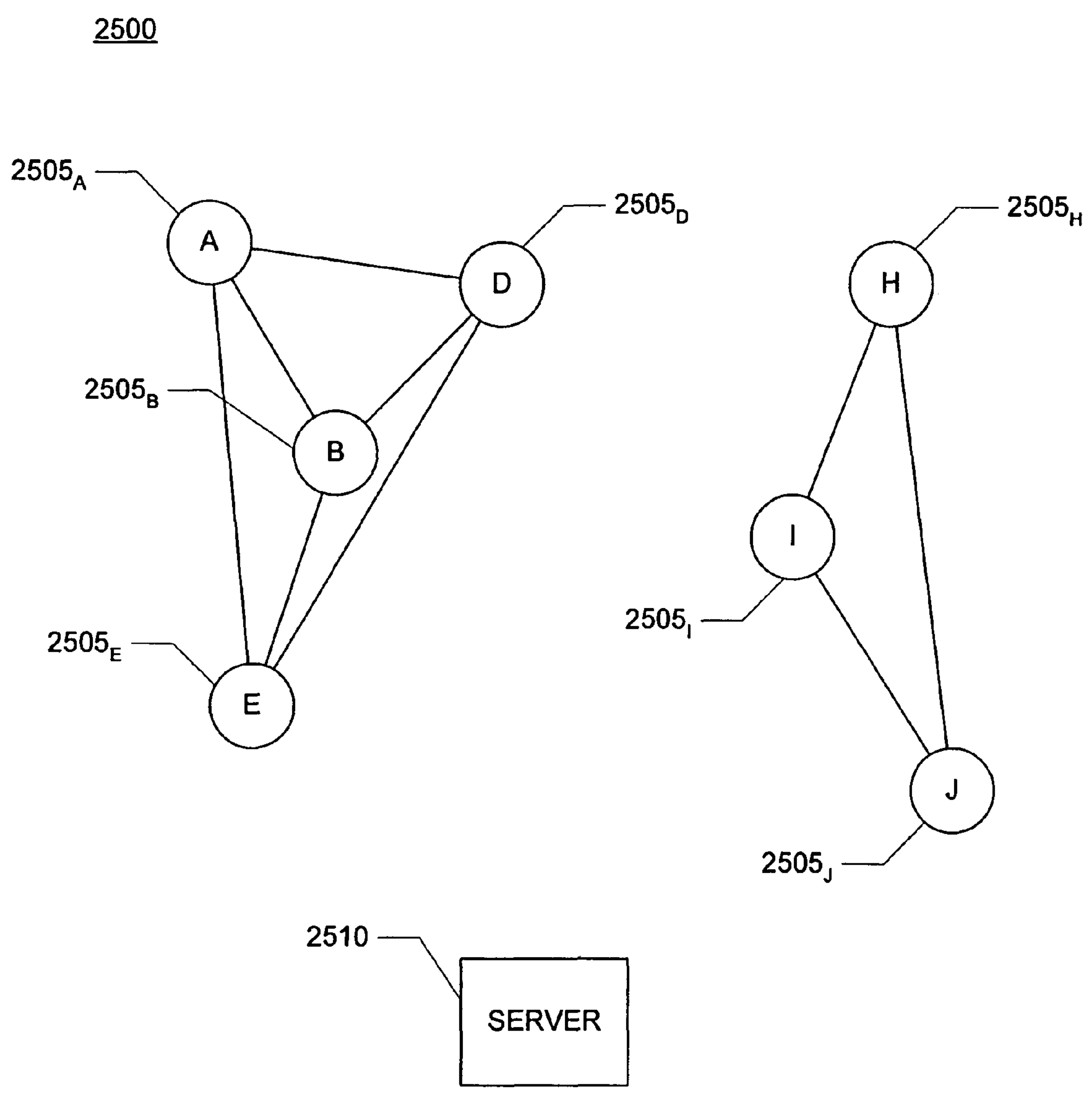

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 26

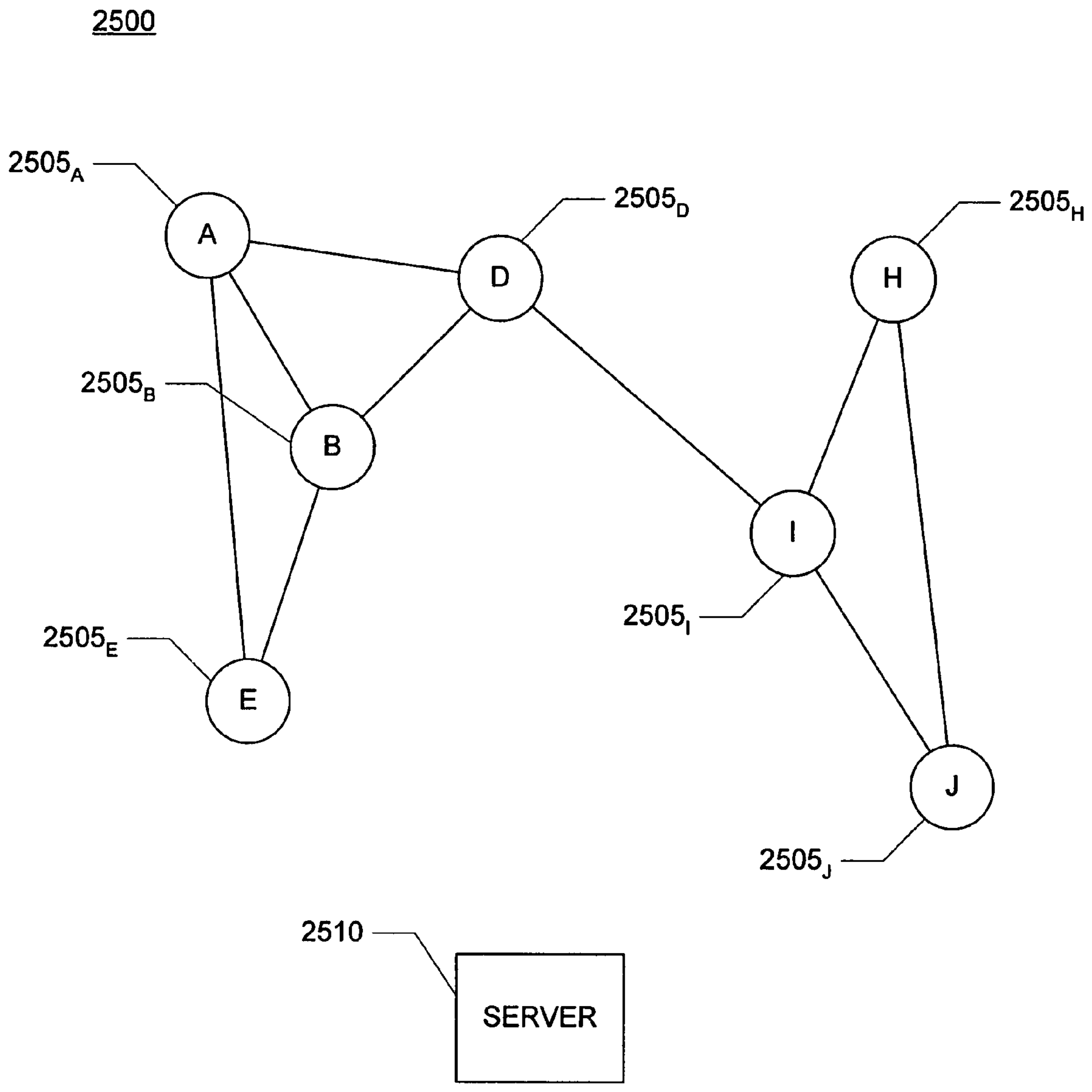

2500
2505A
A
2505D
D
2505H
H
2505B
B
I
2505I
2505E
E
J
2505J
2510
SERVER

FIG. 27

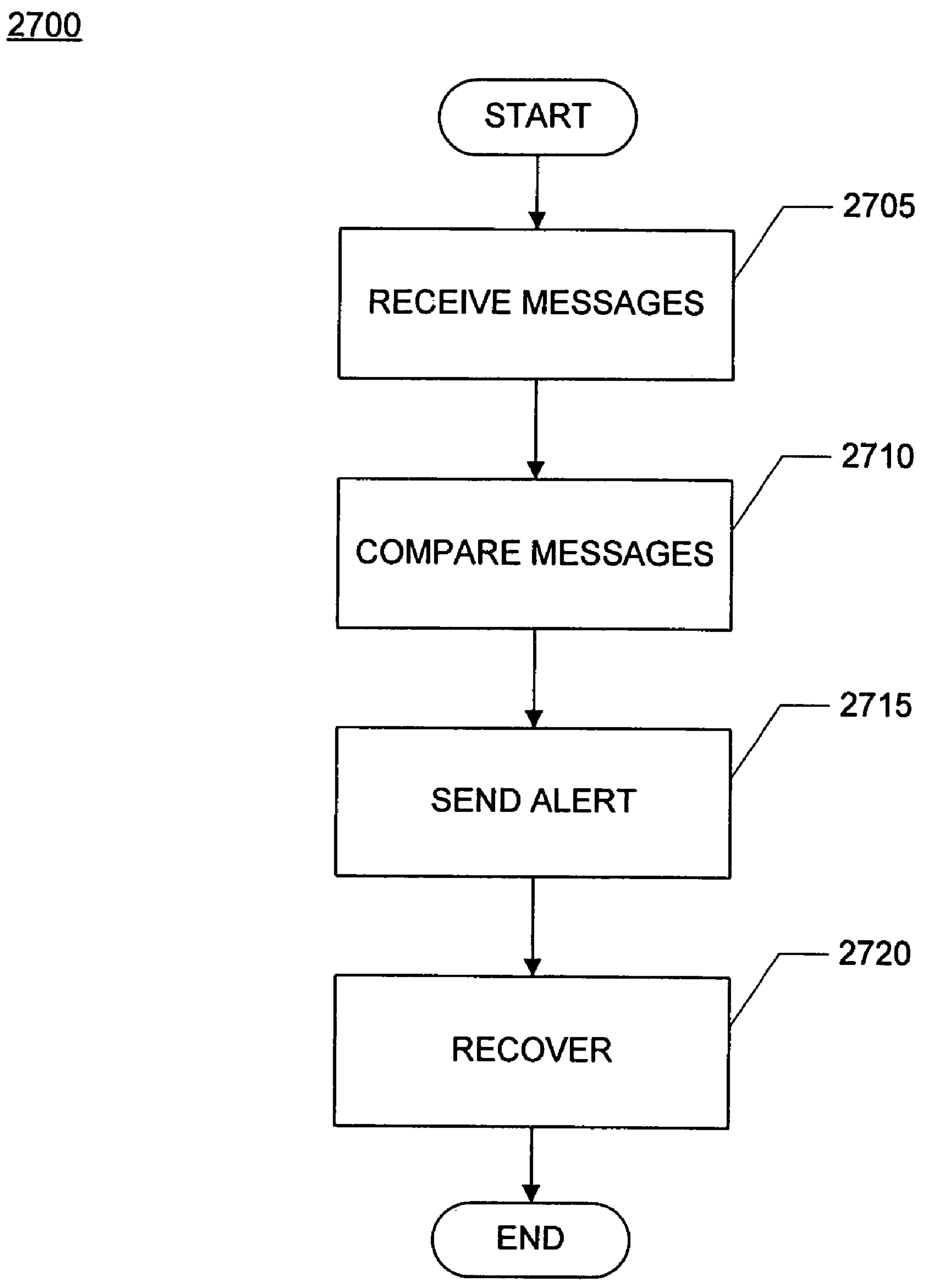

2700
START
RECEIVE MESSAGES
2705
COMPARE MESSAGES
2710
SEND ALERT
2715
RECOVER
2720
END

FIG. 28

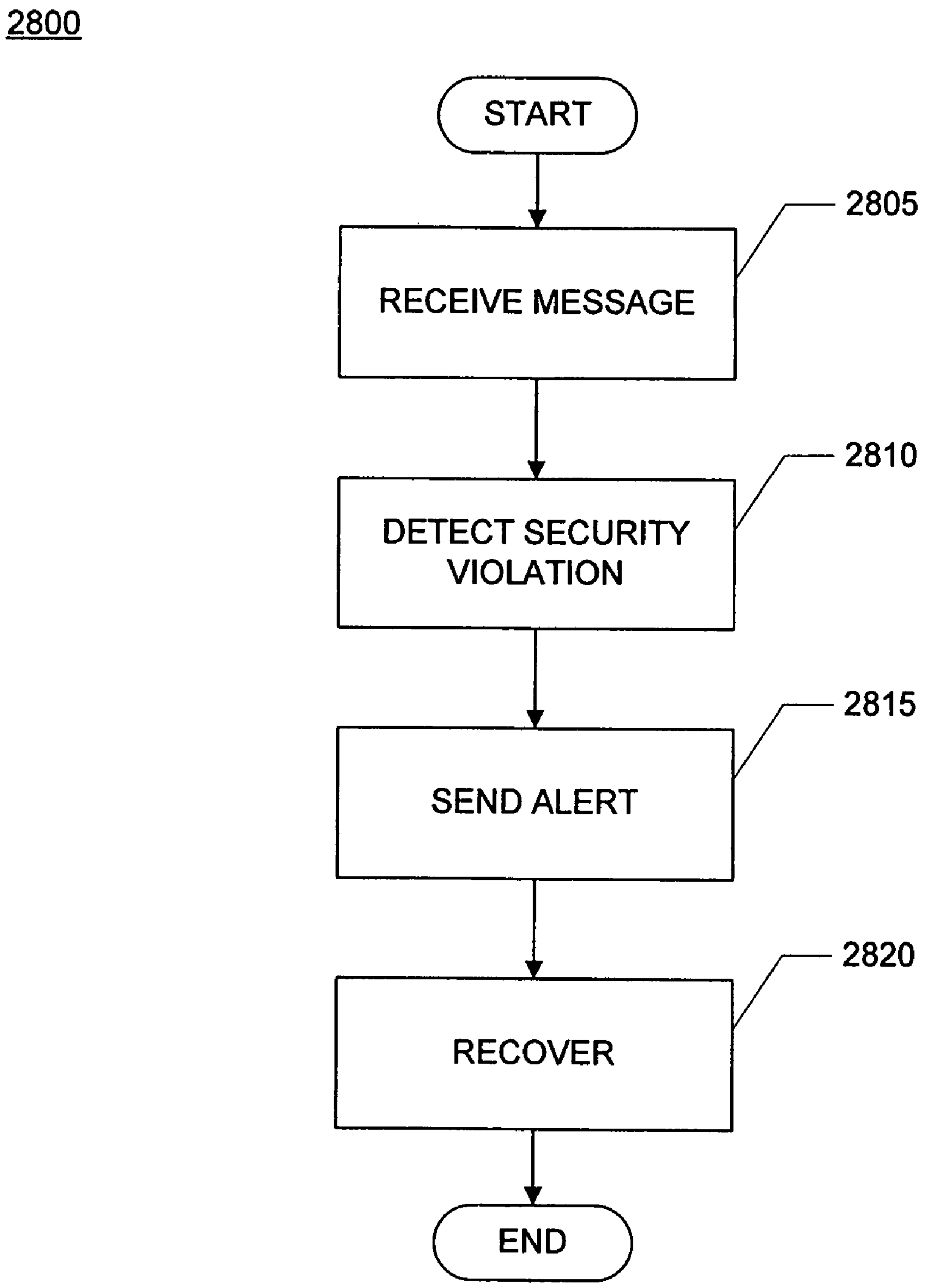

2800
START
RECEIVE MESSAGE
2805
DETECT SECURITY VIOLATION
2810
SEND ALERT
2815
RECOVER
2820
END

FIG. 29

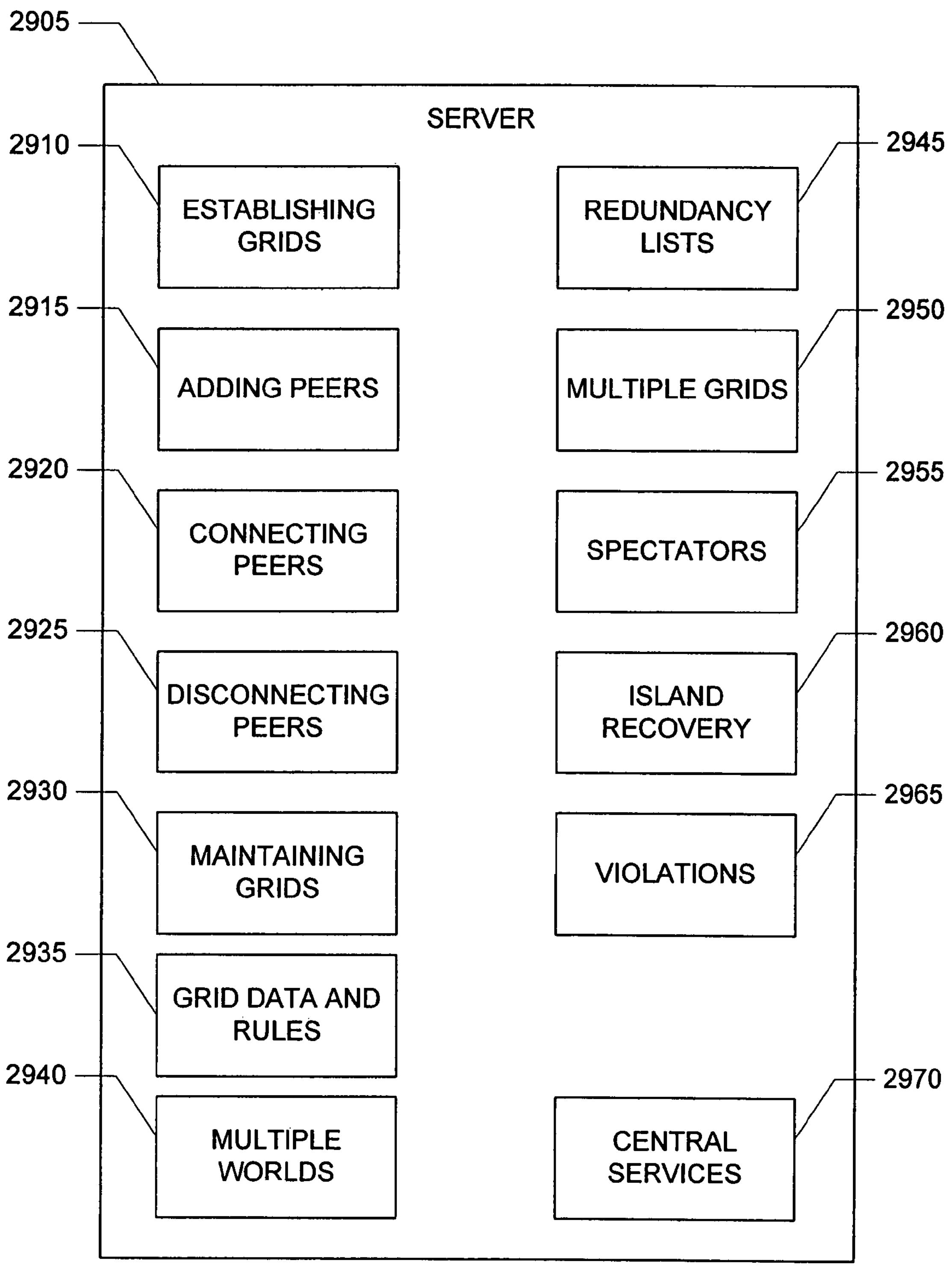

2905
SERVER
2910
ESTABLISHING GRIDS
2915
ADDING PEERS
2920
CONNECTING PEERS
2925
DISCONNECTING PEERS
2930
MAINTAINING GRIDS
2935
GRID DATA AND RULES
2940
MULTIPLE WORLDS
2945
REDUNDANCY LISTS
2950
MULTIPLE GRIDS
2955
SPECTATORS
2960
ISLAND RECOVERY
2965
VIOLATIONS
2970
CENTRAL SERVICES

FIG. 30

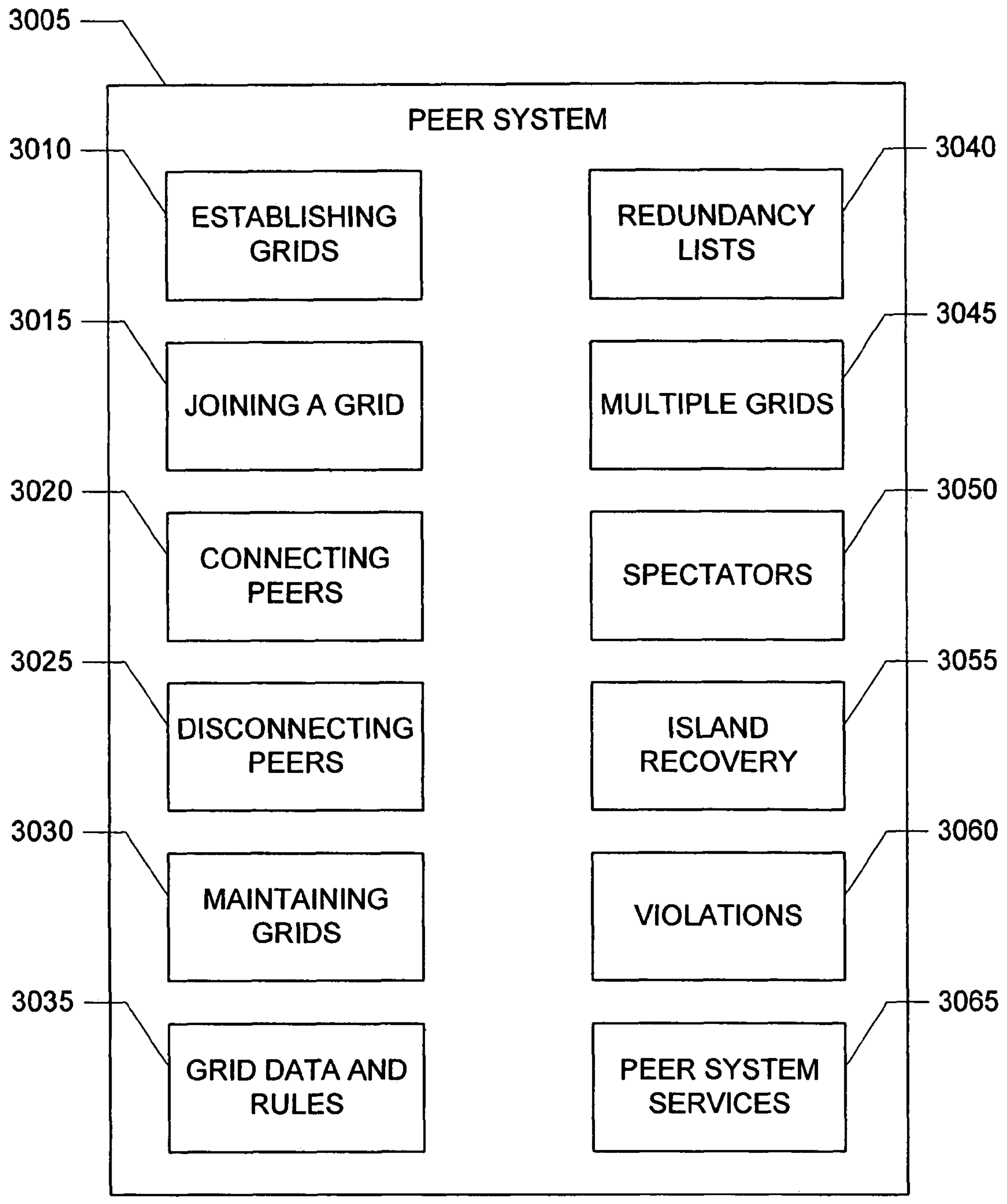

3005
PEER SYSTEM
3010
ESTABLISHING GRIDS
3015
JOINING A GRID
3020
CONNECTING PEERS
3025
DISCONNECTING PEERS
3030
MAINTAINING GRIDS
3035
GRID DATA AND RULES
3040
REDUNDANCY LISTS
3045
MULTIPLE GRIDS
3050
SPECTATORS
3055
ISLAND RECOVERY
3060
VIOLATIONS
3065
PEER SYSTEM SERVICES

FIG. 31A

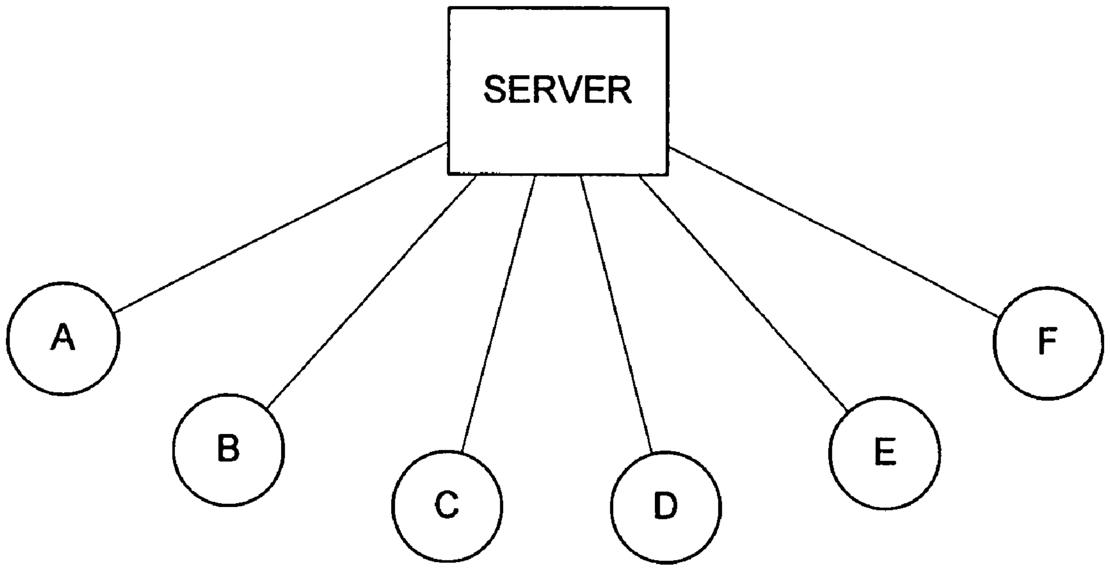
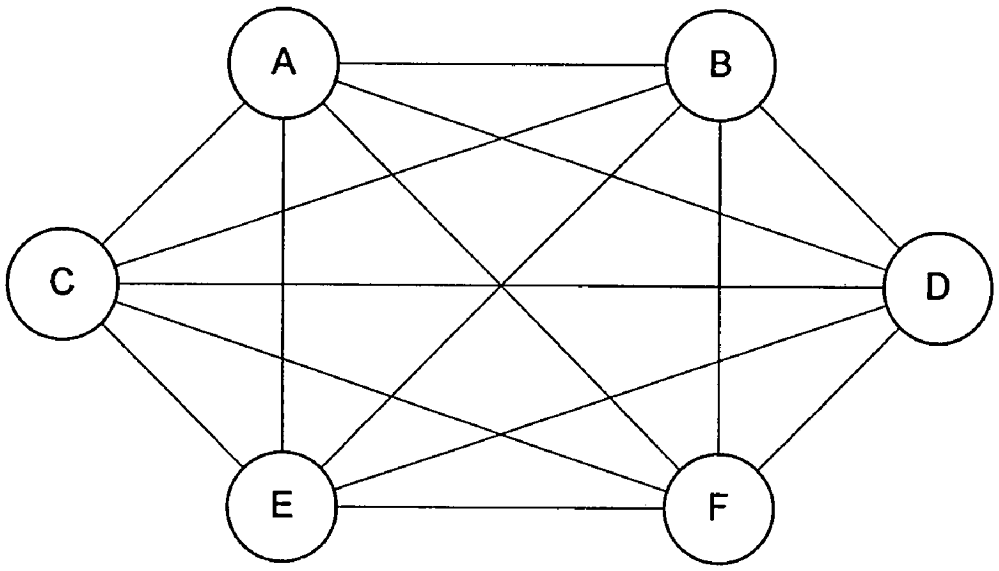


FIG. 31B



PEER-TO-PEER RELAY NETWORK WITH DECENTRALIZED CONTROL

This is a division of application serial number 10/700,798, filed Nov. 3, 2003 now U.S. Pat. No. 7,627,678, which claims benefit to U.S. Provisional Application No. 60/513,098; filed Oct. 20, 2003, the entirety of which is incorporated herein by reference.

BACKGROUND

In a typical client-server network, each of the clients in the network establishes a connection to a central server. A client requests services and data from the server. To communicate with another client, a client sends a request to the server. Typically, the clients do not establish direct connections to one another. In a client-server network with N clients, each client has 1 connection to the server, and the server has N respective connections to each of the clients. For example, as shown in FIG. **31**A, in a client-server network with 6 clients, each client has 1 connection to the server, and the server has 6 respective connections to the clients.

In a typical peer-to-peer network (or "P2P network"), each member (or peer) in the peer-to-peer network establishes a connection to each of the other members. Using these direct peer-to-peer connections, the members send data to and request data from the other members directly, rather than using a centralized server (e.g., compared to a typical client-server network where members interact through the server). Typically, each member in the network has similar responsibilities in the network and the members are considered generally equivalent (as network members). In a peer-to-peer network with N peers, each peer has N-1 connections to other peers. For example, as shown in FIG. **31**B, in a peer-to-peer network with 6 peers, each peer has 5 connections to other peers

In some peer-to-peer networks, a server is also used by the members for some centralized services, such as address discovery (e.g., for establishing the connections for building the peer-to-peer network).

SUMMARY

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a peer-to-peer relay network includes: a plurality of N peer systems; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit, said connection limit is greater than or equal to 2, said connection limit is less than or equal to N-2, and each peer system is configured to relay data to peer systems connected to that peer system according to a set of one or more relay rules.

In another implementation, a server for a peer-to-peer relay network includes: means for establishing a peer-to-peer relay network; means for adding a peer system to a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for tracking connections in a peer-to-peer relay network.

In another implementation, a peer system for a peer-to-peer relay network includes: means for relaying data to any other peer systems connected to said peer system in a peer-to-peer relay network; means for establishing a peer-to-peer relay network; means for joining a peer-to-peer relay network; means for connecting to another peer system in a peer-to-peer relay network; means for maintaining a peer-to-peer relay network; and means for disconnecting from another peer system connected to said peer system in a peer-to-peer relay network.

In another implementation, a method of relaying data in a peer-to-peer relay network includes: receiving data at a relaying peer system from a sending peer system connected to said relaying peer system in a peer-to-peer relay network; applying a set of one or more relay rules to select zero or more peer systems indicated by said set of one or more relay rules to which to relay said data; and relaying said data to any peer systems selected by applying said set of one or more relay rules.

In another implementation, a method of adding a peer system to a peer-to-peer relay network includes: opening a connection between a server and a joining peer system; providing grid information to said joining peer system indicating one or more established peer-to-peer relay networks; receiving a grid selection from said joining peer system indicating a selected peer-to-peer relay network, wherein said selected peer-to-peer relay network has one or more member peer systems; providing network addresses of each of said one or more member peer systems to said joining peer system; and receiving a connection update from said joining peer system indicating to which member peer systems said joining peer system is connected; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to a connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of joining a peer-to-peer relay network includes: sending a join message from a joining peer system to each of one or more member peer systems in a peer-to-peer relay network; receiving a join response from at least one of said one or more member peer systems, wherein each join response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting one or more member peer systems up to a connection limit according to a set of one or more connection rules; opening a connection with each selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of establishing a peer-to-peer relay network includes: opening a connection between said server and an establishing peer system, wherein the establishing peer system is one of said member peer systems; sending a request to create said peer-to-peer relay network from said establishing peer system to said-server; receiving a creation confirmation at said establishing peer system from said server; wherein said establishing peer system stores a connection limit defining a number of other peer systems up to which said establishing peer system is permitted to connect, and said establishing peer system stores a set of one or more relay rules for relaying data to other peer systems connected to said establishing peer system.

In another implementation, a method of connecting peer systems in a peer-to-peer relay network includes: sending a connection available message from a disconnected peer system to one or more member peer systems in a peer-to-peer relay network when said disconnected peer system has a number of open connections to member systems that is less than a connection limit; receiving a connection available response from at least one of said one or more member peer systems, wherein each connection available response is positive or negative, and a positive join response indicates the sending member peer system has an available connection and a negative join response indicates the sending member peer system does not have an available connection; selecting a member peer system according to a set of one or more connection rules; opening a connection with said selected member peer system; wherein each member peer system is connected to a number of other member peer systems that is less than or equal to said connection limit and each member peer system stores a set of one or more relay rules for relaying data to the other member peer systems connected to that member peer system.

In another implementation, a method of maintaining a peer-to-peer relay network includes: sending a maintenance message from a peer system to each of one or more connected peer systems connected to said peer system in a peer-to-peer relay network; evaluating any responses received from said one or more connected peer systems; and closing the connection between said peer system and a connected peer system if the response from that connected peer system is not acceptable; wherein each peer system is connected to a number of other peer systems that is less than or equal to a connection limit and each peer system stores a set of one or more relay rules for relaying data to the other peer systems connected to that peer system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network.

FIG. **2** shows a block diagram of one implementation of a message.

FIG. **3** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network.

FIG. **4** shows a flowchart of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules.

FIG. **5** shows a flowchart of one implementation of establishing a peer-to-peer relay network.

FIG. **6** shows a flowchart of one implementation of connecting a peer to a peer-to-peer relay network.

FIG. **7** shows a flowchart of one implementation of selecting peers for joining a peer-to-peer relay network.

FIG. **8** shows a flowchart of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network.

FIG. **9** shows a flowchart of one implementation of disconnection in a peer-to-peer relay network.

FIG. **10** shows a flowchart of one implementation of maintaining a peer-to-peer relay network.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

FIG. **19** shows a flowchart of one implementation of building a redundancy list in a peer-to-peer relay network.

FIG. **20** shows a flow chart of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network.

FIG. **21** shows a flow chart of one implementation of relaying a message from a peer system that belongs to multiple grids.

FIG. **22** shows a flow chart of one implementation of relaying a message in a grid supporting spectators and participants.

FIG. **23** shows a flow chart of one implementation of detecting islands in a grid.

FIG. **24** shows a flow chart of one implementation of removing islands in a peer-to-peer relay network.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands.

FIG. **27** shows a flow chart of one implementation of detecting a cheating violation in a peer-to-peer relay network.

FIG. **28** shows a flow chart of one implementation of detecting a security violation in a peer-to-peer relay network.

FIGS. **29** and **30** show block diagrams of one implementation of a server and a peer system, respectively.

FIGS. **31**A and **31**B illustrate typical client-server and peer-to-peer architectures.

DETAILED DESCRIPTION

The present invention provides methods and apparatus for implementing peer-to-peer relay. In one implementation, a plurality of computer systems is connected to form a peer-to-peer network. Each computer system is connected to up to a predetermined number of other computer systems. To communicate, a computer system sends a message to each of the connected systems. When a computer system receives a message from another computer system, the receiving computer system sends or-relays the message to other computer systems according to the relay procedures or rules for that peer-to-peer relay network. Following the relay rules, the messages propagate throughout the network to all the member computer systems.

FIG. **1** shows a representation of one implementation of a peer-to-peer relay network **100**. A peer-to-peer relay network can also be referred to as a "grid." In FIG. **1**, a group of 10 peer systems **105**$_{A \ldots J}$ (also referred to as "peers") are connected to form a peer-to-peer relay network. Each peer system **105** is a network-enabled game console, such as a PlayStation 2™ game console with a network adapter, as offered by Sony Computer Entertainment Inc. The peer systems **105** are connected directly (e.g., wired or wireless connections) or indirectly (e.g., through an intranet or a public IP network such as the Internet). In one implementation, the peer systems **105** are connected using UDP or TCP connections. The peer systems **105** exchange data to support a network environment or activity, such as a chat environment or an online game.

Each peer **105** also has a connection to a central server **110**, such as a UDP or TCP connection through the Internet (the connections to the server **110** are not shown in FIG. **1**). The server **110** is a server computer system providing centralized services to the connected peer systems **105**. In one implementation, the server provides an address directory of peer systems and tracks which peer systems are connected with which. Examples of other server services include, but are not limited to: authentication, player matching, and tracking peer system addresses. As described below, in some implementations, the server can support multiple independent or related peer-to-peer relay networks. In one implementation, the server supports multiple environments or worlds, dividing or grouping clients into the environments and filtering data appropriately. In one implementation, the server includes one or more aspects of the servers described in co-pending and commonly assigned U.S. patent applications Ser. No. 11/211075 ("Configuration Switching: Dynamically Changing Between Network Communication Architectures"), filed Jul. 31, 2002, and Ser. No. 10/359359 ("Multi-User Application Programming Interface"), filed Feb. 4, 2003, the disclosures of which are incorporated herein by reference. In another implementation, the peers do not use a centralized server (e.g., building the grid through direct communication and relaying data).

The network **100** has a connection limit of **3**. The connection limit is set by the server and defines the maximum number of connections each peer **105** is permitted to have in the grid. In another implementation, one peer (e.g., the peer establishing the grid) sets or multiple peers negotiate the connection limit. In FIG. **1**, the connection limit is 3 and each peer **105** has 3 connections. Peer systems A-J each have 3 connections to other peers (peer system **105**$_A$ is also referred to as peer system A or peer A). The network **100** is a 3-connection peer-to-peer relay network so each peer **105** has 3 connections to other peers.

The peers **105** communicate by broadcasting messages throughout the network **100**. The peers **105** propagate the messages by relaying received messages to connected peers **105** according to the relay rules of the network **100**. In this implementation, the relay rules define that a peer **105** relays a message to each of the peers **105** connected to the peer **105**, with two exceptions: (i) a peer **105** does not relay a message that the peer **105** has already relayed, and (ii) a peer **105** does not relay a message back to the peer **105** from which the relaying peer **105** received the message. In one implementation, a peer **105** also does not relay a message to a peer **105** from which the relaying peer **105** has already received the message (e.g., when the relaying peer **105** receives the message from multiple peers **105** before the relaying peer **105** has relayed the message). In other implementations, different or additional rules can be used. The relay rules (and other rules) are established by the server or are pre-set in the peer systems (or their system software). In another implementation, the rules can be modified dynamically, such as by propagating messages with rule updates throughout the grid.

In one application of the network **100**, the peers **105** are playing a network game. In the course of the game, a peer **105** generates an update message reflecting actions or events caused by the peer **105**. For example, during the execution of the game software on a player's computer system (e.g., peer A), the computer system generates update data to be used by other players' computer systems representing actions in the game such as moving or shooting (e.g., updating the position of a player). For the update to be effective, each of the peers **105** needs to receive the update from the updating peer **105**. The peers **105** relay the update messages throughout the network **100** to propagate the message to each peer **105**.

In one example, peer A has an update to send to the other peers. Peer A builds an update message including the update data, an identifier indicating peer A is the source of the update, and a sequence identifier to differentiate this message from others sent out by peer A and provide a relative sequence. Peer A sends the message to its connected peers: B, C, D. Peer B sends the message received from peer A to peers D and E. Peer B does not send the message to peer A because peer B received the message from peer A. Similarly, peer C sends the message from peer A to peers G and H, and peer D sends the message from peer A to peers B and G. When peer B receives the message from peer D, peer B does not relay the message again because peer B recognizes that this is the same message (using the identifiers of the message). Similarly, peer D does not relay the message received from peer B. Assuming that the connections between peers are substantially the same in terms of the amount of time to transfer a message between peers, in the next set of relays, peer E relays the message from peer B to peers F and I, peer G relays the message from peer C to peers D and F (or relays the message from peer D to peers C and F, depending on which message arrived at peer C first), and peer H relays the message from peer C to peers I and J. At this time, every peer has received the update message from peer A. However, peers F, I, and J have just received the message, so these peers will relay the message. Peer F relays the message from peer E to peers G and J (or from peer G to peers E an J, whichever arrived first), peer I relays the message from peer E to peers H and J (or from peer H to peers E and J, whichever arrived first), and peer J relays the message from peer H to peers F and I. By this time, all of the peers have sent or relayed the message once. Because the peers will not relay the same message again, the propagation of this message ends.

In this way, the message propagates throughout the peer-to-peer network **100**. This propagation of update information among the peer systems **105** participating in the game supports the game and game environment. The peer systems **105** can distribute data throughout the network **100** without using the centralized server **110** for distribution. In addition, each peer **105** is not directly connected to every other peer **105**, saving resources. As a result, the grid **100** limits each peer's network bandwidth requirement (since it only needs to communicate with a limited number of other clients) while allowing data from any single client to quickly spread to every other peer in the grid (e.g., using UDP sockets).

In other implementations, a peer-to-peer relay network includes more or less peer systems and the network has a different connection limit. Depending upon the number of peers, the connection limit, and the rules for establishing connections, not all peers may have all their connections filled and so there may be a peer (or more) with an available connection.

In another implementation, the connection limit can vary. In one implementation, the connection limit is specific to each peer system, with some, all, or none of the peers having different connection limits. Each peer sets its connection limit, or is assigned a connection limit by a server. In one example, peers X and Y each have a connection limit of 5, and peer Z has a connection limit of 4, and the remaining peers each have a connection limit of 3. In another implementation, the connection limit is dynamic. In this case, the server adjusts the connection limit for the peers, such as based on network performance (e.g., when network traffic is low, the connection limit is low). In another implementation, one or more of the peer systems each adjust their respective connection limit dynamically. Alternatively, the server adjusts the connection limit for specific peer systems dynamically (e.g., adjusting some but not all).

FIG. **2** shows a block diagram of one implementation of a message **205**. The message **205** is built by a peer system to be sent to other peers in a peer-to-peer relay network. For example, referring to FIG. **1**, when peer A has an update message to send to the other peers, peer A builds a message such as the message **205**. The message **205** includes: addressing data **210**, an origin identifier **215**, a sequence value **220**, and payload data **230**. The addressing data **210** includes network addressing information to send the message **205** from the peer to another peer. In one implementation, the addressing data **210** includes an IP address for the sending peer and an IP address for the intended recipient peer. The origin identifier **215** identifies the peer that built the message **205**. This identifier **215** indicates to peers throughout the peer-to-peer relay network the origin of the message propagating through the network. Using the origin identifier **215**, a peer receiving the message **205** can determine from which peer in the network the message **205** originated. The sequence value **220** identifies the specific message **205** and provides relative sequence information. Using the sequence value **220**, a peer receiving the message **205** can determine whether a particular message has already been received and can determine the order or sequence of messages sent from the peer indicated by the origin identifier **215**. The data **230** is the payload data for the message **205**. For an update message (e.g., in a game), the payload data **230** is the update data to be used by the recipient peers. In alternative implementations, different types of messages can be used, and messages with different formats from that shown in FIG. **2** can be used (e.g., including different or additional information). For example, a message can include a file or part of a file or frame of data such as a frame of game data or a frame or part of an audio file being published to the members of the grid. The receiving peers could reconstruct the whole file using the sequence value included in each of the messages. In another example, a message includes additional identification information, such as an identifier indicating to which grid the message belongs for relaying by peers belonging to multiple grids.

FIG. **3** shows a flowchart **300** of one implementation of a peer relaying a message in a peer-to-peer relay network. Initially, the peer is connected to one or more other peer systems in a peer-to-peer relay network.

The peer receives a message from a sending peer through a connection between the peer and the sending peer, block **305**. The message includes an origin identifier, a sequence value, and payload data (e.g., update data), as in the message shown in FIG. **2**.

The peer selects connections to which to relay the received message, block **310**. The peer selects the connections from the available connections of the peer according to the relay rules for the peer-to-peer relay network. After applying the relay rules, the peer may have selected some, none, or all of the peer's connections.

The peer relays the message to each of the selected connections, block **315**. The peer builds a message for each selected connection. For each message to send, the peer uses the received message but updates the addressing information as appropriate (e.g., changing the sender to the peer and the recipient to the recipient peer for the connection). Accordingly, the payload data remains the same. In another implementation, a peer can also add data to the message or change data in the message. The peer sends the built messages to the appropriate recipients.

FIG. **4** shows a flowchart **400** of one implementation of a peer relaying a message in a peer-to-peer relay network according to a set of relay rules. The relay rules used in FIG. **4** are an example of one set of relay rules. Other implementations can use different or additional relay rules. Initially, the relaying peer is connected to N other peer systems in a peer-to-peer relay network. For example, in the network shown in FIG. **1**, peer D is connected to 3. other peers (and so N=3 in this case). The relay rules for FIG. **4** for relaying a message are:

1. Do not relay the message twice
2. Do not relay the message back to the sender
3. Do not relay the message to the origin peer
4. Relay the message to the peers on the connections available after applying rules 1 and 2

The relaying peer receives a message, block **405**. The relaying peer determines whether the relaying peer has already received this message, block **410**. The relaying peer compares identification data for the message with data stored by the relaying peer for messages already received. In one implementation, each peer maintains a received message table of origin identifiers and sequence values for messages that have been received. The relaying peer retrieves the origin identifier and sequence value from the received message and compares this information with data stored in the relaying peer's received message table. If the relaying peer determines that the relaying peer has previously received this received message (e.g., the peer finds an entry in the received message table storing the origin identifier and sequence value of the received message), the relaying peer does not relay the received message. In another implementation, the relaying peer checks to determine if the relaying peer has previously relayed the received message.

If the relaying peer determines that the relaying peer has not previously received this message, the relaying peer records that the message has been received, block **412**. In one implementation, the relaying peer adds an entry to the relaying peer's received message table for the origin identifier and sequence value of the received message. If the table already has an entry for this origin identifier and sequence value, the relaying peer does not change the table.

After recording that the message has been received, the relaying peer sets a counter, block **415**. The relaying peer uses the counter to step through each of the relaying peer's available connections. In one implementation, the relaying peer sets an integer counter i to 1.

The relaying peer determines whether the relaying peer received the message from the peer connected to the connection indicated by the counter, block **420**. The received message includes addressing information indicating the sender of the received message. The counter indicates a connection and so indicates a connected peer and that peer's addressing information. For example, peer D in FIG. **1** has 3 connections and peer D has assigned a number to each connection: peer A is connected to connection **1**, peer B is connected to connection **2**, and peer G is connected to connection **3**. So, when the counter i is 1, peer D checks to see if the received message was sent by peer A by comparing the addressing information (the sender) for the received message with the addressing information for peer A stored by peer D. If the received message was sent to the relaying peer by the peer connected to the connection indicated by the counter the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter, the relaying peer determines whether the peer connected to the connection indicated by the counter is the origin peer system for the received message, block **422**. The received message includes information indicating the peer that is the origin of the received message (the peer that generated the data of the message originally, recall the origin identifier **215** of FIG. **2**). If the peer connected to the connection indicated by the counter is the origin peer system for the received message the relaying peer does not relay the message to that peer.

If the received message was not sent to the relaying peer by the peer connected to the connection indicated by the counter and the peer connected to the connection indicated by the counter is not the origin peer system for the received message, the relaying peer relays the message to that connected peer, block **425**. The relaying peer builds a message for the indicated connection. The relaying peer makes a copy of the received message and updates the addressing information as appropriate (e.g., changing the sender to be the relaying peer and the recipient to be the connected peer connected to the indicated connection). Accordingly, the payload data remains the same. The relaying peer sends the built messages to the connected peer through the indicated connection.

The relaying peer determines whether all the connections have been checked, block **430**. The relaying peer compares the counter to the number of connections established by the relaying peer in the peer-to-peer relay network. For example, the relaying peer compares the counter i to the value of N (the number of connections held by the relaying peer). If the relaying peer has checked all the connections, the relaying peer has completed relaying for this received message.

If the relaying peer has not checked all the connections, the relaying peer increments the counter, block **435**. For example, the relaying peer sets the counter i to be i+1. After incrementing the counter, the relaying peer determines whether the relaying peer received the received message from the peer connected to the connection indicated by the incremented counter, returning to block **420**.

As noted above, in other implementations, different, additional, or fewer relay rules can also be used. In one implementation, the relaying peer does relay the message back to the sender (e.g., so the sender can confirm that the relaying peer did not change the data). In another implementation, the relaying peer does not relay the message to the peer that is indicated as the origin of the message (e.g., as indicated by the origin identifier of the message). In another implementation, the relaying peer does not relay the same message to the same connected peer again. In another implementation, the relaying peer selects a subset of the available connections to relay the message, such as selecting the peers with the lowest and highest response times. In another implementation, each peer relays the message to all the peer's connected peers subject to a hop count stored in the message so that the message will only be relayed a certain number of times. In another implementation, a peer relays the same message a limited number of times (more than once).

FIG. **5** shows a flowchart **500** of one implementation of establishing a peer-to-peer relay network. Initially, a peer system and a server are deployed, such as peer A and the server **110** in FIG. **1**. The peer system opens a connection to the server, block **505**. The peer system is connecting to the server to establish a peer-to-peer relay network (or grid) and can be referred to as an "establishing peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further. The peer system submits a create grid request to the server, block **510**. The create grid request indicates the peer's identification information and that the peer is requesting the server to establish a new peer-to-peer relay network. In one implementation, the request also includes conditions that the peer requests the server to apply (e.g., restrictions on joining the grid). In another implementation, the request indicates a connection limit and a set of rules for use in the grid (e.g., relay rules and connection rules). The server registers the new grid, block **515**. The server maintains tables or lists of data tracking the established grids. The server creates a new table for the new grid and adds the requesting peer to the table. The server sends confirmation to the peer that the grid has been established, block **520**. The confirmation includes any identification or access information the peer needs to access the grid. In one implementation, the confirmation includes the connection limit and the rules for the grid (e.g., relay rules).

FIG. **6** shows a flowchart **600** of one implementation of connecting a peer to a peer-to-peer relay network. Initially, a peer-to-peer relay network has been established by a peer and server, such as peer A and the server **110** in FIG. **1**. A peer system connects to the server, block **605**. The peer system is connecting to the server to join a peer-to-peer relay network (or grid) and can be referred to as a "new peer" or "joining peer." The connection to the server can be direct or an indirect network connection. In one implementation, the peer is assigned to or joins and registers in a subsection of the space

or one of multiple worlds or environments maintained by the server. The server authenticates the peer before allowing the peer to interact further.

The peer selects a grid from the available grids of the server, block **610**. In one implementation, the peer requests a list of available grids and selects from that list. In another implementation, the server supplies the list of available grids automatically when the peer connects to the server. In one implementation, the server provides a list of available grids for the world in which the peer has registered. The server can also provide additional information to assist in the selection (e.g., which peers are already members of each grid). The peer submits the grid selection to the server.

The server sends the addresses of the peers that have already joined the selected grid, block **615**. The addresses indicate how to communicate with the grid members (e.g., IP addresses). The addresses are for establishing peer connections with the grid members; not connections through the server. If the selected grid has restricted access and the new peer is not permitted to join the selected grid, the server does not provide the addresses to the peer and offers to let the peer select a different grid. In one implementation, the server provides the connection limit and rules for the selected grid with the addresses to the new peer.

The new peer sends a join message to each of the grid members, block **620**. The join message indicates the address of the new peer and that the peer is new to the grid. In another implementation, the new peer sends a connection available message indicating the peer's address and the number of connections the peer has available (similar to when a peer loses a connection, as described below). In another implementation, the new peer sends a join message to one grid member and that grid member begins to relay the join message through the grid.

The grid members receive the join message and each sends a join response back to the new peer, block **625**. A join response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. The new peer receives the join responses.

The new peer selects which of the grid members to which to connect, block **630**. The new peer uses a set of connection rules to select peers for connection. For example, in one implementation, the new peer selects from the peers sending positive responses a number of peers up to the connection limit for the grid in the order the positive responses were received by the new peer (e.g., for a connection limit of 3, the new peer selects the peers corresponding to the first three positive responses received). Different implementations can use different sets of connection rules. The new peer stores the response times for each of the selected peers. In another implementation, the new peer stores the response times for all the responses (positive and negative).

After selecting the peers for connection, the new peer opens connections to the selected peers, block **635**. The new peer sends a connection request to each of the selected peers and the selected peers confirm the request, opening the connections (unless connections have become unavailable for the selected peers). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet). In one implementation, when peers open a connection, each peer informs the server of the connection.

In another implementation, the server facilitates joining the grid by forcing one or more connections. The server can cause one peer to close a connection and open a connection to another indicated peer. The server can also cause a peer to close one or more of its connections.

FIG. **7** shows a flowchart **700** of one implementation of selecting peers for joining a peer-to-peer relay network, such as in block **630** of FIG. **6**. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers.

The new peer selects the peer corresponding to the first received positive response, block **705**. This positive response was received before the others and represents the fastest available connection. The new peer selects the peer corresponding to the last received positive response, block **710**. This positive response was received after the others and represents the slowest available connection. To determine which response is last, the new peer waits until all responses have been received or for a defined period of time and then declares the last received in that period to be the last. The new peer randomly selects peers from the remaining positive responses until the new peer has selected a number of peers equal to the connection limit, block **715**. These selections support an even distribution of fast and slow connections through the grid.

As noted above, in various implementations, different or additional connection rules can be used. In one implementation, the new peer selects the peers for the first and last positive responses and then selects the peers corresponding to positive responses in increasing order of response time (after the first). In another implementation, the new peer selects peers as the responses arrive (e.g., reserving one space for the last received positive response), rather than waiting to begin selecting peers. In another implementation, the new peer selects peers using a response time threshold (e.g., do not select peers with a response time above some limit). In another implementation, the new peer selects peers based on characteristics of the peers (using information provided in the join responses), such as storage capacity, processing speed, access levels, or available functions.

In one implementation, a peer system classifies the connections according to the selection process used for selecting those connections. For example, a peer stores information indicating which of the open connections corresponds to the join response received with the lowest response time and which of the open connections corresponds to the join response received with the highest response time. As connections are adjusted for peers disconnecting and new peers joining the grid, the peer can adjust the stored classifications of connections.

In another implementation, the new peer uses the server to assist in opening connections. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The new peer sends the join messages directly to the indicated grid members.

If there are fewer positive responses than the connection limit, the new peer will have remaining available connections. In one implementation, the new peer can force another peer to close an established connection and open a connection with the new peer.

FIG. **8** shows a flowchart **800** of one implementation of forcing a peer to give a connection to a new peer in a peer-to-peer relay network. Initially, a new peer has selected a grid and sent out join messages to the member peers of that grid. The new peer has received join responses back from the member peers. However, after selecting the peers for all the positive responses, the new peer still has available connections.

The new peer selects a peer corresponding to a negative response, block **805**. The new peer selects a negative response using the same connection rules for positive responses (e.g., the first received negative response according to the rules from FIG. **7**). Alternatively, the new peer uses a different set of force connection rules. The new peer does not select a peer to which the new peer is already connected.

The new peer sends a force connection request to the selected peer, block **810**. The force connection request indicates that the new peer has at least one available connection (or specifically how many) and that the recipient peer is to open a connection with the new peer.

The new peer receives the force connection request and selects a connection to close, block **815**. The recipient peer selects a connection to close using the connection rules in reverse. For connection rules based on response time, the recipient peer uses the stored response times from join responses (and connection available responses, as described below). In one implementation, to select among randomly selected peers, the recipient peer selects the last peer selected, or again randomly selects a peer. In another implementation, the recipient peer uses a different set of forced disconnection rules.

The recipient peer closes the selected connection, block **820**. The recipient peer sends a close message to the peer connected to the selected connection and the two peers close the connection. The peer connected to the selected connection now has an available connection and sends out a connection available message to the grid, as described below.

The recipient peer sends a confirmation to the new peer, and the two peers open a new connection, block **825**. The new peer now has one less available connection. If the new peer has more available connections, the new peer repeats the process, returning to block **805** to select another negative response.

In another implementation, the new peer does not force another peer to open a connection unless the new peer has at least two available connections. Alternatively, a different threshold can be used (e.g., three). In another implementation, the new peer sends a force connection message when the new peer does not have at least some number of connections (a connection floor).

In another implementation, the recipient peer for a force connection message has the option to decline (e.g., depending on network load balancing). If declined, the new peer selects another peer to which to send a new force connection message.

In another implementation, if a new peer has two or more available connections and is sending a force connection message, the new peer includes information in the message indicating that the new peer has two available connections. When the recipient peer has selected a connection to close, the recipient peer indicates to the connected peer for the selected connection (the remote peer) that the new peer has another available connection (and includes the address of the new peer if appropriate). After the recipient peer has closed the connection with the remote peer, the remote peer sends a connection available message directly to the new peer (unless the new peer is already connected to the remote peer). The new peer opens a new connection with the recipient peer (selected by the new peer) and another new connection with the remote peer (selected by the recipient peer). In this way, the new peer can quickly establish two connections. If the new peer still has another two available connections, the new

peer can again send a force connection message indicating two available connections to another selected recipient peer.

When a peer system disconnects from another peer system, each of the peers then has an available connection. If one (or both) of these peers is still in the grid (i.e., has not disconnected from the grid), the peer sends out a connection available message to the peer's remaining connected peers to be relayed through the grid to all the other peers in the grid.

FIG. **9** shows a flowchart **900** of one implementation of disconnection in a peer-to-peer relay network. Initially, a peer system (the disconnected peer) is connected to at least two other peer systems in a peer-to-peer relay network.

The disconnected peer becomes disconnected from one of the peers to which the disconnected peer was initially connected, block **905**. The disconnection can occur because of a voluntary disconnection on either end or a failure in the connection itself (e.g., part of the path between the peers fails). For example, a voluntary disconnection can occur when the peer determines that a connected peer is non-responsive (as described below) or when the peer is forced to open a connection with a new peer (as described above). In one implementation, the server can cause a peer to close one or more connections resulting in corresponding disconnections.

The disconnected peer sends a connection available message to the peers remaining connected to the disconnected peer, block **910**. The connection available message indicates that the disconnected peer now has an available connection. In another implementation, the connection available message indicates the number of connections the peer has available.

The peers connected to the disconnected peer relay the connection available message, block **915**. The peers in the grid send connection available responses back to the disconnected member, block **920**. A connection available response indicates whether the responding peer has any available connections or not. A positive response indicates that the responding peer has an available connection. A negative response indicates that the responding peer does not have an available connection. The responding peers record the new peer's address from the join message and use that address to send the join responses. Alternatively, the responding peers send the responses back through the grid to be relayed to the disconnected peer. The disconnected peer receives the connection available responses.

The disconnected peer selects one of the grid members to which to connect, block **925**. The disconnected peer uses the connection rules to select a peer for connection, but the disconnected peer does not select a peer to which the disconnected peer is already connected. For example, in one implementation, the disconnected peer uses the response times of the connection available responses and the stored response times of the peers still connected to the disconnected peers to select a peer to replace the lost connection. Different implementations can use different sets of connection rules. The disconnected peer stores the response time for the selected peer. In another implementation, the disconnected peer stores the response times for all the responses (positive and negative). In one implementation, the disconnected peer does not select a peer from which the disconnected peer has disconnected within a certain time period.

After selecting a peer for connection, the disconnected peer opens a connection to the selected peer, block **930**. The disconnected peer sends a connection request to the selected peer and the selected peer confirms the request, opening the connection (unless the connection has become unavailable for the selected peer). The connections between peers can be direct or indirect (e.g., across a network, such as the Internet).

In one implementation, the connected peers send an update to the server confirming the connection.

Similar to the implementation described above for joining a grid referring to FIG. **8**, in one implementation, if the disconnected peer still has an available connection after attempting to open a connection using a connection available message (e.g., because all the connection available responses were negative), the disconnected peer can send out a force connection message, as described above.

In another implementation, the disconnected peer uses the server to assist in opening a new connection. In one implementation, the server provides a list of grid members with available connections and those member peers' addresses. The disconnected peer sends the connection available messages directly to the indicated grid members.

The peer systems in the grid maintain the grid by periodically polling one another. In one implementation, connected peers send each other messages periodically to confirm the connection and the connected peer is still functioning.

FIG. **10** shows a flowchart **1000** of one implementation of maintaining a peer-to-peer relay network. Initially, multiple peer systems are connected in a grid.

A peer sends a maintenance message to each of the peers connected to that peer, block **1005**. The maintenance message is a request for the recipient to provide a confirmation that the maintenance message was received. In one implementation, the peer sends a ping message (or pings) each connected peer. The peer evaluates the responses received to the maintenance messages, block **1010**. The peer determines whether the responses are satisfactory or not. In one implementation, if a response is not received from a connected peer, the peer determines that the connection for the peer has failed (either because of the connection or because of the connected peer). If a response is not received before a time limit has expired, the peer determines that the connection for the peer has failed. The peer closes the connections for any connections the peer has determined have failed, block **1015**. The peer sends a close connection request to the connected peer on a failed connection. When the peer receives confirmation, the peer closes the connection. If the peer cannot communicate with the connected peer on a failed connection or does not receive confirmation within a time limit, the peer closes the connection without confirmation. In another implementation, a peer waits to close a connection until the connection has been noted as failed for a period of time or number of failures. In one implementation, the peer sends an update to the server confirming any closed connections.

If the peer has closed any connections, the peer has voluntarily disconnected from one or more peers and sends out appropriate connection available messages (e.g., as described above referring to FIG. **9**).

In another implementation, the peers use the server to evaluate failed connections. For example, when a peer determines that a connection has failed, the peer sends a request to the server for assistance. The server sends a message to the peer at the other end of the failed connection to confirm whether the peer has failed or the connection failed. The server then informs the peers to facilitate opening new connections or adjusting the network as appropriate.

FIGS. **11-18** illustrate an example of one implementation of building, adjusting, and maintaining a grid.

In FIG. **11**, a peer system **1105**$_A$ (peer A) has established a peer-to-peer relay network (grid) **1100** using a server **1110** (the connection between peer A and the server **1110** is not shown). The connection limit for this grid is 3, so peer A has three available connections. In FIG. **12**, a second peer system **1105**$_B$ (peer B) has joined the grid **1100**. When peer B joins,

peer B sends a join message to peer A and peer A sends a positive join response to peer B. Peer A and peer B open a connection.

In FIG. **13**, two more peer systems **1105**$_C$ and **1105**$_D$ (peer C and peer D) have already joined the grid **1100**. Each of the four grid members peers A-D has established three connections with the other peers in the grid **1100**. A new peer system **1105**$_E$ (peer E) joins the grid. However, when peer E sends a join message to the other peers, all the join responses are negative because each of peers A-D already have the maximum number of connections permitted by the connection limit for the grid **1100**. In FIG. **14**, peer E has forced a connection to be opened. Peer E selects peer B from among the negative responses (e.g., because peer E received peer B's response first) and sends a force connection message to peer B. Peer B selects peer D to close a connection and closes the connection with peer D. Peer B confirms the connection with peer E and peers B and E open a new connection. When peer B closes the connection with peer D, peer D has an available connection. Peer D sends a connection available message to peers A and C and the peers relay the message throughout the grid **1100**. Peers A, B, and C do not have available connections and so send negative responses to peer D. Peer E has two available connections and sends a positive response to peer D. Peer D opens a connection with peer E. Peer E still has an available connection and so sends out a connection available message. However, all the responses are negative. Peer E has two established connections and only has one available connection, so peer E does not force another connection to be opened.

In FIG. **15**, peer A disconnects from the grid **1100**. Peer A was connected to each of peers B, C, and D. When peer A disconnects, peers B, C, and D each have an available connection. Peers B, C, and D send out connection available messages and peers B, C, D, and E each send positive responses. After evaluating the responses to the connection available responses and eliminating peers for already existing connections, the peers B-E establish connections as shown in FIG. **16**. Each of peers B-E now has three connections.

In FIG. **17**, three new peer systems **1105**$_F$, **1105**$_G$, and **1105**$_H$ (peers F, G, and H) have joined the grid **1100** and established connections. As part of the regular activity to maintain the grid, the peers B-H each send ping messages to their connected peers. For example, peer B pings peers D, E, and G on a regular basis. Peer D does not provide a satisfactory response to peer B for peer B's ping message (e.g., the response from peer D is too slow or does not arrive at peer B). In FIG. **18**, peer B has closed the connection peer D. When peer B closes the connection, peer B and peer D have available connections. Peers B and D send out connection available messages to be relayed through the grid **1100**. Peer B receives positive responses from peers G and D. Peer B is already connected to peer G so will not select peer G for a new connection. Peer B just disconnected from peer D for a failed connection and so will not select peer D for a new connection. Peer B does not open a new connection (peer B has two open connections and only available connection, so peer B does not attempt to force a connection, though in another implementation peer B may). Peer D receives positive responses from peers B and G. Peer B just disconnected from peer D for a failed connection so peer D will not select peer B for a new connection (or peer B would refuse a new connection request). Peer D selects peer G and opens a connection to peer G.

In the examples illustrated in FIGS. **11-18**, the peers of the grid **1100** open and close connections to build and adjust the grid without relying on the server **1110** to manage the connections (though the server **1110** does assist in providing a new peer with the addresses of the current member peers of a grid).

Redundancy Lists

In one implementation, the peers in a grid reduce redundant message traffic by avoiding sending messages determined to be redundant based on current paths in the grid.

In this implementation, each peer in the peer-to-peer relay network stores a redundancy list. The redundancy list of a peer indicates other peers to which the peer will not send messages that originated from a designated peer. Accordingly, each entry in the redundancy list indicates an origin peer and a destination peer (connected to the relaying peer). When a peer receives a message that indicates an originating peer that is in the peer's redundancy list, the peer will not relay that message to the connected peer indicated by the corresponding entry in the redundancy list. In another implementation, the peers can turn on and turn off the redundancy list functionality (e.g., at the request of a server, such as after determining a security problem has arisen).

FIG. **19** shows a flowchart **1900** of one implementation of building a redundancy list in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network. A recipient peer is connected to at least two other peers.

The recipient peer receives a redundant message from connected peer, block **1905**. The redundant message is redundant because the recipient peer has already received the same message. The recipient peer identifies the redundant message as being the same using information in the received message. As described above, in some implementations, each peer maintains a list of messages received to avoid relaying the same message twice. The recipient peer can also use this list to recognize a redundant message.

The recipient peer builds a redundancy update message, block **1910**. The recipient peer includes in the redundancy update message the information identifying the origin of the message and information identifying the recipient peer. For example, the recipient peer retrieves the origin identifier from the redundant message (e.g., recall the message shown in FIG. **2**) and stores the origin identifier in the redundancy update message.

The recipient peer sends the redundancy update message to the sender of the redundant message, block **1915**. The redundant message includes in its address information address information for the sender of the redundant message.

The sender of the redundant message receives the redundancy update message and updates the redundancy list for the sender, block **1920**. The sender retrieves the information from the redundancy update message identifying the origin of the redundant message and the recipient of the redundant message (the recipient peer). The sender adds an entry to the sender's redundancy list indicating that the sender should not send a message originating from the indicated origin to the recipient peer.

For example, referring to the grid **100** shown in FIG. **1**, peer B receives messages originating from peer C from each of peers A, D, and E. Assuming peer B receives the message originating from peer C from peer A first, the messages originating from peer C received from peers D and E are redundant messages. Peer B builds redundancy update messages to send to peers D and E indicating peer C as the origin and peer B as the recipient. Peer B sends the redundancy update message to peer D. Peer D updates its redundancy list to indicate that peer D is not to relay messages originating from peer C to peer B. Peer E receives a similar redundancy update message from peer B and also updates its redundancy list in a similar way.

As peers connect and disconnect to and from the grid, the paths between clients change and so redundancy lists can become inaccurate. Accordingly, when a peer disconnects from the grid, the remaining peers update redundancy lists.

FIG. **20** shows a flow chart **2000** of one implementation of updating redundancy lists for a disconnecting peer in a peer-to-peer relay network. Initially, multiple peers systems are connected to form a peer-to-peer relay network. A disconnecting peer is connected to at least two other peers.

The disconnecting peer disconnects from the grid, block **2005**. The peers previously connected to the disconnecting peers are now disconnected peers. Each of the disconnected peers follows the same process below.

The disconnected peer builds a clear redundancy message, block **2010**. The clear redundancy message indicates information identifying the disconnected peer. The disconnected peer sends the clear redundancy message to the peers still connected to the disconnected peer, block **2015**. A peer that receives the clear redundancy message from the disconnected peer updates its redundancy list, block **2020**. The peer receiving the clear redundancy message removes entries in the peer's redundancy list affecting relaying messages to the disconnected peer indicated by the clear redundancy message.

Returning to the example described above referring to FIGS. **1** and **19**, peer D has an entry in its redundancy list indicating that peer D should not relay messages originating from peer C to peer B. If peer A disconnects from the grid, peer B recognizes the disconnection of peer A and builds a clear redundancy message. Peer B sends a clear redundancy message to peers D and E. Peer D receives the clear redundancy message from peer B and clears the entry in peer D's redundancy list indicating that peer D should not relay messages originating from peer C to peer B. Accordingly, the next time that peer D receives a message originating from peer C, peer D will once again relay message to peer B. Peer E updates its redundancy list similarly.

Multiple Grids

In one implementation, a peer system can belong to multiple peer-to-peer relay networks. Each grid can be related or independent. The connections established according to each grid can be independent. Accordingly, a peer can be connected to one peer in one grid but not in another (even though the two peers are both in both grids). In one implementation, if two peers are connected in two grids, the peers use a single connection. A message includes information indicating to which grid the message belongs. A peer relays a received message according to the connections established corresponding to the indicated grid for the message.

In one implementation, the members of a peer-to-peer relay network can create sub-networks within the peer-to-peer relay network. In this case, each of the members of a sub-network is also a member of the larger grid. For example, a peer-to-peer relay network includes all the players in a game as peer systems and each team (including sub-sets of the total players) has a sub-network of peer systems (e.g., for private communication in the game). In this way, the peers can establish a multi-channel environment for desirably distributing and receiving data.

In another implementation, the peer-to-peer relay networks are independent but share one or more member peer systems. For example, a group of peers can establish a grid to support a lobby or chat environment and another group of peers including at least one peer of the first group can establish a grid to support a particular game. In another example, a group of peers form a grid for a clan (organization) and some of those peers join or create other grids to play games.

For example, in an online environment, all the peers in the environment are connected to a single main grid. The main grid is for general announcements and general services. Peers create, join, and leave additional smaller grids to access online services such as chat rooms or games. Peers can use the main grid to communicate before a smaller grid has been established, such as when a new peer wants to join a grid (rather than using a server). Because all the control messages can be broadcast through the main grid, every peer can independently maintain a list of available grids and a list of active peers in each grid. In one implementation, the peers do not use a centralized server.

FIG. **21** shows a flow chart **2100** of one implementation of relaying a message from a peer system that belongs to multiple grids. Initially, multiple peers systems are connected to form two peer-to-peer relay networks. A relaying peer is a member of both grids, and has respective connections and relay rules for each grid.

The relaying peer receives a message, block **2105**. The message includes a grid identifier indicating to which grid the message belongs.

The relaying peer selects the grid indicated by the received message, block **2110**. Each grid has a respective set of connections and a respective set of relay rules. By selecting a grid, the relaying peer selects a set of connections to use and a set of relay rules to use for relaying the received message.

The relaying peer selects connections according to the selected grid and the corresponding relay rules, block **2115**. Using the relay rules for the selected grid, the relaying peer select any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2120**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received message to indicate the received message is being relayed from the relaying peer to the selected peer.

Spectators

In one implementation, the peers in a grid are classified as participants or spectators. A participant peer generates new messages to be relayed throughout the grid. A spectator peer does not generate new messages and acts as a pass-through node in the grid. Both participants and spectators relay messages to their connected peers according to the relay rules of the grid. In some applications, there may be many spectators for each participant. In one implementation having multiple participants, each participant has a connection to at least one other participant.

In one example, a group of participants play an online game while spectators watch (observing data without changing the game data). The number of spectators can be very large (e.g., thousands). Other examples include performances (e.g., music), speeches, and teaching. In some applications, because the peers handle distribution by relaying data, the load on a server for distribution does not always increase as the number of spectators increases.

In one implementation, when a peer joins a grid, the peer joins the grid as a participant or as a spectator. If the peer joins the grid as spectator, the peer is not authorized to create new messages and send the new messages into the grid to be relayed throughout the grid. If a spectator generates a new message and sends the new message to the peers connected to the spectator, the peers receiving the new message from the spectator will not forward or relay the received message. In one implementation, some or all of the spectators could form another related grid as participants (e.g., to discuss a game being watched in the first grid).

FIG. **22** shows a flow chart **2200** of one implementation of relaying a message in a grid supporting spectators and participants. Initially, multiple peers systems are connected to form a peer-to-peer relay network supporting participants and spectators. Each of the peers systems stores a list of the peers that are participants. In one implementation, the participant peers periodically broadcast messages indicating which peers are participants. In another implementation, the server facilitates identifying the participants.

A relaying peer receives a message, block **2205**. The message includes an origin identifier indicating the peer that created the message.

The relaying peer confirms that the origin of the received message is a participant peer, block **2210**. The relaying peer stores a list of participant peers. The relaying peer compares the peer identified as the origin of the received message with the list of participant peers. If the origin peer for the received message is not a participant (i.e., is a spectator), the relaying peer does not relay the received message.

If the origin peer for the received message is a participant, the relaying peer selects connections according to the relay rules for the grid, block **2215**. Using the relay rules, the relaying peer selects any appropriate connections for relaying the received message.

The relaying peer sends the received message to the selected peers, block **2220**. Before relaying the message, the relaying peer adjusts the received message for each selected peer, such as by updating the address information for the received-message to indicate the received message is being relayed from the relaying peer to the selected peer.

In another implementation, the spectators are not in the same grid as the participants. The spectators form a parallel spectator grid linked to the participant grid. The spectators receive data from the participants and relay the data in the spectator grid. The link(s) between the grids can be provided by a server or gateway, or by connections between selected peers from each grid.

In another implementation, a spectator can be a conditional spectator. A conditional spectator can request permission to generate data to be relayed throughout the grid. If the spectator has received permission, the spectator can send a message that the peers in the grid will relay (e.g., the message includes an authorization flag). The permission can be granted by a server, by a selected peer as a moderator, or by the participants (one or more). For example, in a teaching environment, the participant is the lecturer and the spectators can request permission to ask questions that will be relayed to all the peers.

Island Recovery

In one implementation, the server and peers in a peer-to-peer relay network support adjusting connections in the grid to avoid or recover from the formation of islands. An isolated group of peers in a grid is referred to as an island. Islands can form in a grid when multiple peers disconnect substantially simultaneously. In the disconnection process described above, the remaining peers send messages indicating available connections, however, with multiple concurrent disconnections, the remaining peers may form isolated groups in the grid. Peers in one island cannot send messages to peers in another island because there is no peer-to-peer connection between the islands. The server detects the formation of islands and interacts with peers to remove the islands.

FIG. **23** shows a flow chart **2300** of one implementation of detecting islands in a grid. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid. When the peers open and close connections, or become disconnected, peers inform the server-for the grid of the changing

connections. In this way, the server tracks all of the connections in the grid. The server also maintains an ordered list of the peers in the grid.

The server sets an island counter, block **2305**. The island counter represents the number of islands. In one implementation, the server sets a counter i to be 1.

The server selects a starting peer, block **2310**. When the island counter is one, the server selects the first peer in the ordered list of peers as the starting peer. When the island counter is greater than one, the server selects as the starting peer the most recently found unmarked peer (as described below).

The server marks each peer connected to the starting peer as belonging to the same island as the starting peer, block **2315**. The server marks peers connected directly to the starting peer and connected indirectly to the starting peers through other peers (e.g., progresses from the starting peer to connected peers and peers connected to those connected peers and so on). The server marks a peer with the current value of the island counter to indicate to which island the peer belongs.

After marking all of the peers connected to the starting peer, the server determines if there is an unmarked peer remaining in the grid, block **2320**. In one implementation, the server progresses through the ordered list of peers searching for an unmarked peer.

If the server finds an unmarked peer, the server increments the island counter, block **2325**. The server increments the island counter to indicate that an additional island has been detected. After incrementing the island counter, the server returns to block **2310** and uses the found unmarked peer as the starting peer.

If the server does not find an unmarked peer, the server determines the number of islands detected, block **2330**. The server has incremented the island counter for each detected island, and so the island counter represents the number of islands detected. If the island counter is equal to one, a single island has been found and so the grid is not divided into multiple islands. If the island counter is greater than one, multiple islands have been found and the grid is divided into islands.

FIG. **24** shows a flow chart **2400** of one implementation of removing islands in a peer-to-peer relay network. Initially, multiple peers systems are connected in a peer-to-peer relay network or grid. The grid has become divided into two islands of peers, where the peers in one island do not have a connection path to the peers in the other island. The server has detected the two islands, such as by using the process shown in FIG. **23**.

The server selects a peer from each island, block **2405**. The server can select the first island peer and the second island peer in various ways. In one implementation, the server selects a peer that has an available connection. In another implementation, the server selects a peer from an island at random.

If the first island peer does not have available connections, the server sends a close connection message to the first island peer to close a connection, block **2410**. The first island peer receives the message from the server and selects a connection to close in the same way as a peer selects a connection to close when receiving a force connection message, as described above. The first island peer closes a connection and so has an available connection.

The server sends an initiate force connection message to the first island peer, block **2415**. The initiate force connection message includes the address of the second island peer. The first island peer receives the message from the server and sends a force connection message to the second island peer.

The second island peer receives the force connection message from the first island peer, selects a connection to close, and closes the selected connection, block **2420**. The second island peer selects the connection to close in the same way as described above for the recipient of a force connection message. If the second island peer has an available connection before closing a connection, the second island peer does not close any of its connections.

The first island peer sends an open connection request to the second island peer, and the two peers open a connection, block **2425**. Once the connection is open, the islands have been joined, forming a single island. The peers send updates to the server confirming the connection. If additional islands remain, as detected as described above, the server returns to block **2405** to connect two more of the remaining islands.

FIGS. **25** and **26** illustrate an example of detecting islands and joining islands. In FIG. **25**, a grid **2500** similar to the grid **1100** in FIG. **11** has been divided into two islands from the simultaneous disconnection of peers C, G, and F. The first island includes peers A, B, D, and E. The second island includes peers H, I, and J. In FIG. **26**, the server has caused peer D to open a connection with peer I, joining the two islands.

Security

In one implementation, the peer-to-peer relay network supports the detection of and recovery from cheating violations or security violations, or both. Cheating violations involve the manipulation of data to change an outcome in the processing of online activity, such as to affect the course of a game. Security violations involve unauthorized data or improper use of data to damage the grid or cause the grid to fail.

FIG. **27** shows a flow chart **2700** of one implementation of detecting a cheating violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from each of its connected peers, block **2705**. As described above, the peers in the grid relay messages throughout the grid. A peer will receive the same message (the same content data, though the address information may be different) through each of its connections with other peers. For example, if a peer has three open connections, the peer receives the same message three times from three respective peers. The peer identifies the messages as being the same message using information in the message indicating the origin and a sequence value, such as the origin identifier **215** and sequence value **220** shown in the message **205** in FIG. **2**. The same message from different peers will have the same origin and sequence information.

The peer compares the messages received from each of the connected peers, block **2710**. The peer compares the data portion of the message, such as the data **230** shown in the message **205** in FIG. **2**. The peer determines if the data portion of the message is different for any of the received messages. In one implementation, if the data portion for a message received from one connected peer is different from the data portion for the same message received from the other connected peers, the peer determines that a cheating violation has occurred. The peer also determines that the one peer that sent the message with the different data is responsible for the cheating violation. Alternatively, the peer uses a different technique to detect a cheating violation or identify the peer responsible for the cheating violation. The peer does not relay the message having a different data portion, if appropriate.

If a cheating violation has occurred, the peer sends a cheating alert, block **2715**. The cheating alert indicates a cheating violation has occurred and which peer is responsible for the cheating violation. The peer sends the cheating alert to the connected peers to relay the alert throughout the grid. In another implementation, the peers send the cheating alert to the server for appropriate handling.

When the peers receive the cheating alert, the peers take action to recover against the violation, block **2720**. The peers take action to prevent the cheating peer from continuing to influence the grid activity. In one implementation, the peers ignore messages from the cheating peer. In another implementation, the peers force the cheating peer to disconnect from the grid. The peers also take action to repair the effect of the message including the different data, such as by sending out a replacement message with correct data as shown by the data in the other messages used to identify the cheating message. Alternatively, one of the peers estimates the correct data and relays the correct data throughout the grid. In another implementation, the peers respond to the cheating alert by informing the server. In this case, the server addresses the cheating violations such as by disconnecting the peer responsible for the cheating violation.

In another implementation, when a peer sends a message, the recipient relays the message back to the sending peer. The sending peer keeps a copy of the sent message. When the sending peer receives the message back from the recipient, the sending peer compares the data of the sent message with the data of the received message. The peer detects a cheating violation by finding a difference. The peer determines that the recipient modified the message and sends out a cheating alert. In one implementation, recovery or repair actions are not taken for a cheating peer until multiple violations have been reported (e.g., as tracked by a server). In another implementation, this send-back check for cheating is a first layer for detecting cheating followed by more complicated procedures once a potential problem has been identified.

In another implementation, the peer detects a cheating violation by comparing the data in a received message with a predicted set of data generated by the peer. If the peer determines that the data in the received message is different from that generated by the peer, the peer determines that the sender of the received message is responsible for a cheating violation and issues an alert.

In an example of detecting a cheating violation in the grid **100** shown in FIG. **1**, peer B receives the same message from each of peers A, D, and E. Peer B identifies the messages as being the same by comparing the origin identifiers and sequence values. If peer B detects that the message from peer A has a different data portion, peer B issues a cheating alert identifying peer A as cheating. Peer B sends the cheating alert to peers D and E (and optionally to peer A). The peers relay the cheating alert until all the peers have received the alert. In response to the alert, the peers will ignore all further messages from peer A. As a result, peers B, C, and D will not relay messages from peer A anymore.

FIG. **28** shows a flow chart **2800** of one implementation of detecting a security violation in a peer-to-peer relay network. Initially, multiple peer systems are connected to form a peer-to-peer relay network or grid.

The peer receives a message from one of its connected peers, block **2805**. The peer analyzes the message and detects a security violation, block **2810**. The peer determines that the message is a security violation by recognizing that the message is invalid or includes invalid data. In another implementation, the peer determines that the message is a security violation by analyzing how the message was sent to the peer. For example, if the message was sent to the peer as one of a large number of repetitions of the same message (e.g. as in a denial of service attack), the peer recognizes that the message is a security violation. In one implementation, a message is sent as a series of packets and the peer detects a security violation at a lower level than a complete message, such as at the packet level. The peer also determines that the sender of the message with the security violation is responsible for the security violation. Alternatively, the peer uses a different technique to detect a security violation or identify the peer responsible for the cheating violation. The peer does not relay a message or data having a security violation.

If a security violation has occurred, the peer sends a security alert, block **2815**. The security alert indicates a security violation has occurred and which peer is responsible for the security violation. The peer sends the security alert to the connected peers to relay the alert throughout the grid. In another implementation, the peer sends the security alert to the server for proper handling.

When the peers receive the security alert, the peers take appropriate action to recover against the violation, block **2820**. The peers take action to prevent the peer violating the security of the grid from continuing to affect or damage the grid. In one implementation, the peers ignore messages from the peer responsible for the security violation. In another implementation, the peers force the peer responsible for the security violation to disconnect from the grid. The peers also take appropriate action to repair any damage caused by the security violation. In another implementation, the peers respond to the security alert by informing the server. In this case, the server addresses the security violation such as by disconnecting the peer responsible for the violation and the action to repair any damage caused to the grid.

FIGS. **29** and **30** show block diagrams of one implementation of a server **2905** and a peer system **3005**, respectively. In other implementations, a server or a peer include fewer components than shown in FIGS. **29** and **30**, or include different or additional components.

The server **2905** operates as described above and includes components to provide the functionality described above, including components for establishing grids **2910**, adding peers **2915**, connecting peers **2920**, disconnecting peers **2925**, maintaining grids **2930**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **2935**, managing multiple worlds **2940**, managing and assisting with redundancy lists **2940**, managing multiple grids **2950**, managing spectators and participants in grids **2955**, handling island detection and recovery **2960**, managing and addressing cheating and security violations **2965**, and central services of the server **2970** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

The peer system **3005** operates as described above and includes components to provide the functionality described above, including components for establishing grids **3010**, joining a grid **3015**, connecting peers **3020**, disconnecting peers **3025**, maintaining grids **3030**, storing and generating grid data (e.g., connections, members, connection limits) and rules (e.g., relay rules, connection rules) **3035**, building, updating, and using redundancy lists **3040**, operating in multiple grids **3045**, operating with and as spectators and participants in grids **3050**, handling island detection and recovery **3055**, managing, detecting, and addressing cheating and security violations **3060**, and peer system services **3065** (e.g., network communication and addressing, player matching, chat facilities, data backup, etc.).

Various implementations of the peer-to-peer relay network provide desirable benefits. A grid can be very useful in a number of network applications, including online massive multi-player computer games. Online game applications are just one example of a larger group of network applications that have one thing in common: sharing and maintaining one common data set. When the data set is updated on one peer, the information is sent to a group of other peers and relayed throughout the grid so each peer will have an updated data set. The relay grid allows connected peers with limited network bandwidth to exchange data among themselves, without going through a central server (for data distribution). This network can be used to exchange game data, other game related information, media files, streaming audio, or streaming video.

For example, in one implementation the peers use the grid for file publishing. A peer in the grid publishes a file (as one message or broken into multiple messages) by sending the file to the peers connected to the publisher and the member peers of the grid relay the file throughout the grid to all the members. In this way all the members of the grid can receive the published file without using a server and without using a direct connection from the published to every peer. In various implementations, any type of file can be published. The files can be data, media, or executable software applications. Examples of files to be published through a grid include, but are not limited to: streaming media (e.g., audio and/or video), media files, replay data from a game or other application, maps, announcements, messages, application data and modules (e.g., a map, a template, a texture, a sound).

The various implementations of the invention are realized in electronic hardware, computer software, or combinations of these technologies. Most implementations include one or more computer programs executed by a programmable computer. For example, in one implementation, each peer system and the server includes one or more computers executing software implementing the peer-to-peer relay network functionality. In general, each computer includes one or more processors, one or more data-storage components (e.g., volatile or non-volatile memory modules and persistent optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices (e.g., mice and keyboards), and one or more output devices (e.g., display consoles and printers).

The computer programs include executable code that is usually stored in a persistent storage medium and then copied into memory at run-time. The processor executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

Various illustrative implementations of the present invention have been described. However, one of ordinary skill in the art will see that additional implementations are also possible and within the scope of the present invention. For example, while the above description describes several implementations of peer-to-peer relay networks discussed in the context of supporting game applications, other applications are also possible, such as file sharing or other data dissemination applications.

Accordingly, the present invention is not limited to only those implementations described above.

What is claimed is:

1. A method of relaying data in a peer-to-peer relay network, comprising:

receiving data at a relaying peer system from a sending peer system connected to said relaying peer system in a peer-to-peer relay network, the sending peer being either an originating peer of the received data or another relaying peer system that is not the originating peer of the received data, wherein the originating peer is a peer that generated the received data originally;

applying a set of one or more relay rules to select zero or more peer systems directly connected to the receiving peer indicated by said set of one or more relay rules to which to relay said data;

determining whether a particular peer system directly connected to the relaying peer system is the originating peer of the received data,

relaying said data to directly connected peer systems other than the originating peer of the received data by applying said set of one or more relay rules; and

comparing information identifying said received data with information stored by the relaying peer system to determine whether said received data has been previously received by the relaying peer system.

2. The method of claim **1**, wherein:

said set of one or more relay rules indicate that the relaying peer system is not to relay to the sending peer system the same data received from the sending peer system.

3. The method of claim **1**, wherein:

said set of one or more relay rules indicate that the relaying peer system is not to relay the same data to the same peer system twice.

4. The method of claim **1**, wherein:

said set of one or more relay rules indicate that the relaying peer system is not to relay the data to the peer system indicated as the origin of the data according to information identifying the received data.

5. The method of claim **1**, further comprising:

storing information identifying the received data.

6. The method of claim **1**, wherein:

said received data is update data for a network environment.

7. The method of claim **1**, wherein:

said received data is update data for an online game.

8. The method of claim **1**, wherein:

at least one peer system is a network-enabled game console.

9. The method of claim **1**, wherein:

at least two peer systems are connected through the Internet.

* * * * *

EXHIBIT 94

US008793315B2

(12) **United States Patent**
Chatani et al.

(10) **Patent No.: US 8,793,315 B2**
(45) **Date of Patent: *Jul. 29, 2014**

(54) **MANAGING PARTICIPANTS IN AN ONLINE SESSION**

(75) Inventors: **Masayuki Chatani**, Foster City, CA (US); **Glen Van Datta**, San Diego, CA (US)

(73) Assignee: **Sony Computer Entertainment America LLC**, San Mateo, 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.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **12/840,977**

(22) Filed: **Jul. 21, 2010**

(65) **Prior Publication Data**

US 2010/0287239 A1 Nov. 11, 2010

Related U.S. Application Data

(63) Continuation of application No. 11/375,526, filed on Mar. 13, 2006, now Pat. No. 7,792,902, which is a continuation of application No. 10/221,128, filed on Jul. 31, 2002, now abandoned.

(60) Provisional application No. 60/381,736, filed on May 17, 2002.

(51) **Int. Cl.**
G06F 15/16 (2006.01)
A63F 9/14 (2006.01)

(52) **U.S. Cl.**
USPC **709/205**; 709/209; 463/42

(58) **Field of Classification Search**
USPC 709/204, 205, 208, 209; 463/62
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,630,757	A	5/1997	Gagin et al.
5,634,129	A	5/1997	Dickinson

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1125617	8/2001
GB	2325543 A	11/1998

(Continued)

OTHER PUBLICATIONS

"Technical Issues of establishing any-to-any 2-way real-time communications over the internet," Apr. 24, 2005, URL http://webarchive.org/web/20050424081036.

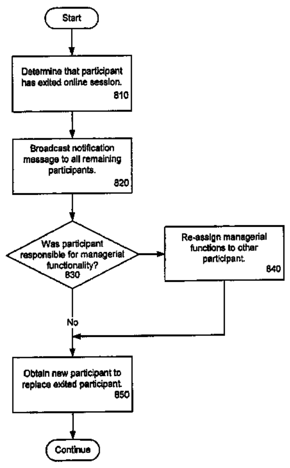
(Continued)

Primary Examiner — Douglas Blair
(74) *Attorney, Agent, or Firm* — Lewis Roca Rothgerber LLP

(57) **ABSTRACT**

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. The application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

9 Claims, 11 Drawing Sheets


Start
Determine that participant has exited online session. 810
Broadcast notification message to all remaining participants. 820
Was participant responsible for managerial functionality? 830
Re-assign managerial functions to other participant. 840
No
Obtain new participant to replace exited participant 850
Continue

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,682,139	A	10/1997	Pradeep et al.	
5,704,032	A	12/1997	Badovinatz et al.	
5,823,879	A	10/1998	Goldberg	
5,826,085	A	10/1998	Bennett	
5,838,909	A	11/1998	Roy et al.	
5,841,980	A	11/1998	Waters et al.	
5,893,106	A	4/1999	Brobst et al.	
5,941,947	A	8/1999	Brown	
5,956,485	A	9/1999	Perlman	
5,984,787	A	11/1999	Redpath	
5,987,376	A	11/1999	Olson et al.	
6,041,312	A	3/2000	Bickerton et al.	
6,050,898	A	4/2000	Vange	
6,106,569	A	8/2000	Bohrer	
6,108,569	A	8/2000	Shen	
6,142,472	A	11/2000	Kliebisch	
6,152,824	A	11/2000	Rothschild et al.	
6,154,782	A	11/2000	Kawaguchi et al.	
6,203,433	B1	3/2001	Kume	
6,219,045	B1	4/2001	Leahy et al.	
6,247,017	B1	6/2001	Martin	
6,311,209	B1	10/2001	Olson et al.	
6,345,297	B1	2/2002	Grimm	
6,352,479	B1	3/2002	Sparks	
6,363,416	B1	3/2002	Naeimi et al.	
6,487,583	B1	11/2002	Harvey	
6,487,678	B1	11/2002	Briskey et al.	
6,519,629	B2	2/2003	Harvey	
6,530,840	B1	3/2003	Cuomo	
6,549,946	B1	4/2003	Fisher	
6,560,636	B2 *	5/2003	Cohen et al.	709/203
6,561,811	B2 *	5/2003	Rapoza et al.	434/236
6,587,874	B1	7/2003	Golla	
6,607,444	B2	8/2003	Takahashi et al.	
6,631,412	B1	10/2003	Glasser et al.	
6,676,521	B1	1/2004	La Mura et al.	
6,748,420	B1	6/2004	Quatrano et al.	
6,761,636	B2	7/2004	Chung et al.	
6,763,371	B1	7/2004	Jandel	
6,799,255	B1	9/2004	Blumenau	
6,931,446	B1	8/2005	Cox et al.	
7,003,550	B1	2/2006	Cleasby et al.	
7,016,942	B1	3/2006	Odom	
7,018,295	B2	3/2006	Sakaguchi et al.	
7,025,675	B2	4/2006	Fogel et al.	
7,056,217	B1	6/2006	Pelkey et al.	
7,089,301	B1	8/2006	Labio et al.	
7,107,312	B2	9/2006	Hackbarth et al.	
7,177,950	B2 *	2/2007	Narayan et al.	709/248
7,188,145	B2 *	3/2007	Lowery et al.	709/214
7,203,755	B2	4/2007	Zhu et al.	
7,290,264	B1	10/2007	Powers et al.	
7,523,163	B2	4/2009	Zhu et al.	
7,587,465	B1	9/2009	Muchow	
7,613,800	B2	11/2009	Dhupelia	
7,711,847	B2	5/2010	Dhupelia	
7,720,908	B1	5/2010	Newson et al.	
7,730,206	B2	6/2010	Newson et al.	
7,792,902	B2 *	9/2010	Chatani et al.	709/205
7,822,809	B2	10/2010	Dhupelia	
7,831,666	B2 *	11/2010	Chatani et al.	709/205
7,877,509	B2	1/2011	Dhupelia	
7,908,393	B2	3/2011	Marr et al.	
7,930,345	B2	4/2011	Dhupelia	
7,962,549	B2	6/2011	Dhupelia	
8,131,802	B2	3/2012	Jacob	
8,560,707	B2	10/2013	Jacob	
2001/0009868	A1	7/2001	Sakaguchi et al.	
2001/0037466	A1	11/2001	Fukutake	
2001/0044339	A1	11/2001	Cordero et al.	
2002/0035604	A1	3/2002	Cohen et al.	
2002/0042830	A1 *	4/2002	Bose et al.	709/230
2002/0049086	A1	4/2002	Otsu	
2002/0062348	A1	5/2002	Maehiro	
2002/0075844	A1	6/2002	Hagen	
2002/0082077	A1	6/2002	Johnson	
2002/0115488	A1	8/2002	Berry	
2002/0133707	A1	9/2002	Newcombe	
2003/0018719	A1	1/2003	Ruths et al.	
2003/0073494	A1	4/2003	Kalpakian et al.	
2003/0190960	A1	10/2003	Jokipii	
2003/0204566	A1	10/2003	Dhupelia	
2003/0204593	A1	10/2003	Brown et al.	
2003/0217135	A1	11/2003	Chatani	
2003/0217158	A1	11/2003	van Datta	
2004/0024879	A1	2/2004	Dingman	
2004/0030787	A1	2/2004	Jandel	
2004/0053690	A1	3/2004	Fogel	
2004/0059711	A1	3/2004	Jandel	
2004/0117443	A1	6/2004	Barsness	
2004/0139228	A1	7/2004	Takeda	
2005/0105526	A1	5/2005	Stiemerling	
2005/0251577	A1	11/2005	Guo	
2005/0259637	A1	11/2005	Chu	
2005/0262411	A1	11/2005	Vertes	
2006/0075127	A1	4/2006	Juncker	
2006/0100020	A1	5/2006	Kasai	
2006/0173958	A1	8/2006	Chatani	
2006/0190540	A1	8/2006	Chatani	
2006/0200551	A1	9/2006	Bali et al.	
2006/0218274	A1	9/2006	Labio et al.	
2006/0218275	A1	9/2006	Labio et al.	
2006/0247011	A1	11/2006	Gagner	
2006/0248144	A1	11/2006	Zhu et al.	
2006/0253595	A1	11/2006	van Datta	
2006/0288103	A1	12/2006	Gobara	
2007/0058792	A1	3/2007	Chaudhari	
2007/0061460	A1	3/2007	Khan	
2007/0076729	A1	4/2007	Takeda	
2007/0165629	A1	7/2007	Chaturvedi	
2007/0174399	A1	7/2007	Ogle et al.	
2007/0191109	A1	8/2007	Crowder	
2007/0208748	A1	9/2007	Li	
2007/0217436	A1	9/2007	Markley	
2008/0280686	A1	11/2008	Dhupelia et al.	
2008/0291839	A1	11/2008	Hooper et al.	
2009/0006545	A1	1/2009	Dhupelia	
2009/0006604	A1	1/2009	Dhupelia	
2009/0077245	A1	3/2009	Smelyansky	
2009/0089363	A1	4/2009	Keohane et al.	
2009/0094370	A1	4/2009	Jacob	
2009/0113060	A1	4/2009	Jacob	
2009/0138610	A1	5/2009	Gobara	
2009/0240821	A1	9/2009	Juncker	
2009/0287828	A1	11/2009	Wei et al.	
2010/0153496	A1	6/2010	Heinla	
2010/0279767	A1	11/2010	Dhupelia	
2010/0285872	A1	11/2010	Dhupelia	
2012/0166651	A1	6/2012	Jacob	

FOREIGN PATENT DOCUMENTS

JP	05-022346	1/1993
JP	63-232725	9/1998
JP	11-234326	8/1999
JP	2000124939	4/2000
JP	2000157724	6/2000
JP	2001187273	7/2001
JP	2001314657	11/2001
JP	2002011251	1/2002
JP	2003099337	4/2003
WO	WO 00/05854	2/2000
WO	WO 00/10099	2/2000
WO	WO 00/68864	11/2000
WO	WO 01/63423	8/2001
WO	WO 01/82678	11/2001
WO	WO 02/35769	5/2002
WO	WO 03/091894	11/2003
WO	WO 03/100643	12/2003
WO	WO 2004/063843	7/2004

(56) **References Cited**

FOREIGN PATENT DOCUMENTS

WO	WO 2005/088466	9/2005
WO	2006/023508 A1	3/2006
WO	WO 2009/045475	4/2009

OTHER PUBLICATIONS

Aronson, Jesse. "Using Groupings for Networked Gaming," Gamasutra.com, Jun. 21, 2000.
Audet, F. NAT Behavioral Requirements for Unicast UDP, Behave Internet-Draft, Jul. 15, 2005.
Boulic, Ronan et al. "Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation," Swiss Fedl Inst. of Tech., Lausanne, Switzerland, Sep. 1997.
"Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Mar. 23, 2007).
Brief for Appellee, In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (May 21, 2007).
Chiueh, Tzi-cker, "Distributed Systems Support for Networked Games," Computer Science Department, State University of New York at Stony Brook, Stony Brook, NY, May 1997.
Cisco Systems, Inc., "Network Flow Management: Resource Reservation for Multimedia Flows," Mar. 19, 1999.
Diot et al., "A Distributed Architecture for Multiplayer Interactive Applications on the Internet," IEEE vol. 13, Issue 4, Aug. 1999.
Festa et al., "Netscape Alumni to Launch P2P Company," Aug. 2, 2001.
Hagsand, O. "Interactive Multiuser VEs in the Dive System" IEEE Multimedia, IEEE Service Center, New York, NY, US vol. 3, No. 1, Mar. 21, 1996, pp. 30-39, XP000582951 ISSN:1070-986X.
Hanada, S. "The Design of Network Game and DirectPlay," Inside Windows, Softbank K.K., vol. 4, No. 4, pp. 42-57, Apr. 1, 1998.
"In Re Masayuki Chatani and Glen Van Datta," United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 11/211,128), Nov. 19, 2007.
"Petition for Panel Rehearing," In Re Masayuki Chatani and Glen Van Datta, Appeal From the United States Patent and Trademark Office, Boar of Patent Appeals and Interferences, In the United States Court of Appeals for the Federal Circuit, 2007-1150 (U.S. Appl. No. 10/211,128), Jan. 3, 2008.
"Reply Brief of Appellants," In re Masayuki Chatani et al., U.S. Court of Appeals for the Federal Circuit (2007-1150) (Jun. 4, 2007).
Rosenberg, J. Interactive Connectivity Establishment (ICE): A Methodology for Netork Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols, Mmusic Internet-Draft, Jul. 19, 2004.
Rosenberg, J. Interactive Connectivity Establishment (ICE): A Methodology for Netork Address Translator (NAT) Traversal for Multimedia Session Establishment Protocols, Mmusic Internet-Draft, Oct. 25, 2004.
Rosenberg, J. Interactive Connectivity Establishment (ICE): A Methodology for Netork Address Translator (NAT) Traversal for Offer/Answer Protocols, Mmusic Internet-Draft, Jul. 17, 2005.
Rosenberg, J. Interactive Connectivity Establishment (ICE): A Methodology for Netork Address Translator (NAT) Traversal for Offer/Answer Protocols, Mmusic Internet-Draft, Jan. 16, 2007.
Rosenberg, J. "Simple Traversal of UDP Through Network Address Translators (NAT)," Behave Internet-Draft, Jul. 17, 2005.
Rosenberg, J. Stun—Simple Traversal of User Datagram Protocols (UDP) Through Network Address Translators (NATs), Network Working Group, Mar. 2003.
Rosenberg, J. Traversal using Relay NAT (TURN), MIDCOM Internet-Draft, Oct. 20, 2003.
Takeda, Y. Symmetric NAT Traversal Using STUN, Internet Engineering Task Force, Jun. 2003.
ECC Report 50. "Technical Issues of Establishing Any-to-Any-2-Way Real-Time Communications over the Internet." Electronic Communications Committee (Ecc). Gothenburg, Jul. 2004.
EP 03 721 1413, European Search Report dated Jun. 30, 2005.
PCT/US03/08682 International Search Report mailed Oct. 14, 2003.
Chinese Application No. 03801033, Office Action mailed Jun. 9, 2006.
Chinese Application No. 03801033, Office Action mailed Jul. 5, 2011.
Chinese Application No. 03801033, Office Action mailed Sep. 25, 2009.
European Application No. 03724201.3, Office Action mailed Jul. 3, 2012.
PCT/US03/12668 International Search Report mailed Jul. 17, 2003.
Chinese Application No. 200810168739.8, Decision of Rejection dated Dec. 11, 2012.
Chinese Application No. 200810168739.8, Office Action dated May 11, 2012.
Chinese Application No. 200810168739.8, Office Action dated May 19, 2011.
Chinese Application No. 20088011547.1, Office Action dated Oct. 12, 2012.
Chinese Application No. 20088011547.1, Office Action dated Mar. 7, 2012.
Chinese Application No. 20088011547.1, Office Action dated Aug. 10, 2012.
EP 08014892.7 Office Action mailed Jul. 20, 2011.
EP 08835745.4 Extended European Search Report dated Jul. 22, 2011.
EP 11004182.9 Extended European Search Report dated Jul. 14, 2011.
EP 11004181.1 Extended European Search Report dated Jul. 22, 2011.
PCT/US08/11415 Search Report and Written Opinion mailed Dec. 5, 2008.
U.S. Appl. No. 10/211,128 Final Office Action mailed Feb. 2, 2004.
U.S. Appl. No. 10/211,128 Office Action mailed Nov. 10, 2003.
U.S. Appl. No. 11/375,526 Office Action mailed Apr. 8, 2008.
U.S. Appl. No. 11/375,526 Final Office Action mailed Jul. 3, 2007.
U.S. Appl. No. 11/375,526 Final Office Action mailed Mar. 2, 2007.
U.S. Appl. No. 11/375,526 Office Action mailed Oct. 24, 2006.
U.S. Appl. No. 11/403,623 Office Action mailed Jun. 25, 2009.
U.S. Appl. No. 11/403,623 Office Action mailed Apr. 9, 2008.
U.S. Appl. No. 11/403,623 Final Office Action mailed Jul. 3, 2007.
U.S. Appl. No. 11/403,623 Final Office Action mailed Mar. 5, 2007.
U.S. Appl. No. 11/403,623 Office Action mailed Oct. 24, 2006.
U.S. Appl. No. 10/359,359 Final Office Action mailed Nov. 27, 2009.
U.S. Appl. No. 10/359,359 Office Action mailed Mar. 31, 2009.
U.S. Appl. No. 10/359,359 Office Action mailed Aug. 27, 2007.
U.S. Appl. No. 10/359,359 Final Office Action mailed Feb. 9, 2007.
U.S. Appl. No. 10/359,359 Office Action mailed Aug. 8, 2006.
U.S. Appl. No. 12/839,306 Office Action mailed Nov. 12, 2010.
U.S. Appl. No. 12/839,311 Office Action mailed Nov. 12, 2010.
U.S. Appl. No. 12/218,581 Office Action mailed Sep. 2, 2010.
U.S. Appl. No. 12/218,581 Office Action mailed Feb. 1, 2010.
U.S. Appl. No. 12/218,581 Office Action mailed Oct. 2, 2009.
U.S. Appl. No. 12/218,591 Office Action mailed Feb. 25, 2009.
U.S. Appl. No. 12/049,954 Final Office Action mailed Dec. 14, 2010.
U.S. Appl. No. 12/049,954 Office Action mailed Jun. 24, 2010.
U.S. Appl. No. 12/235,438 Final Office Action mailed Jan. 4, 2012.
U.S. Appl. No. 12/235,438 Office Action mailed Aug. 8, 2011.
U.S. Appl. No. 12/235,438 Final Office Action mailed Aug. 31, 2010.
U.S. Appl. No. 12/235,438 Office Action mailed Apr. 15, 2010.
U.S. Appl. No. 13/412,361 Final Office Action mailed Nov. 28, 2012.
U.S. Appl. No. 13/412,361 Office Action mailed Jul. 30, 2012.

* cited by examiner

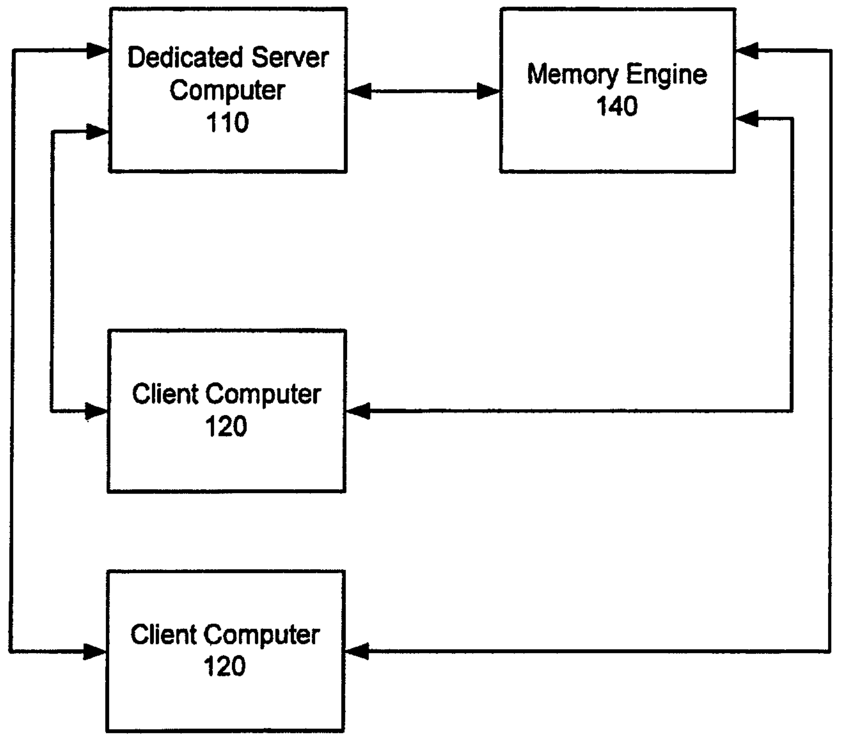


Figure 1

PRIOR ART

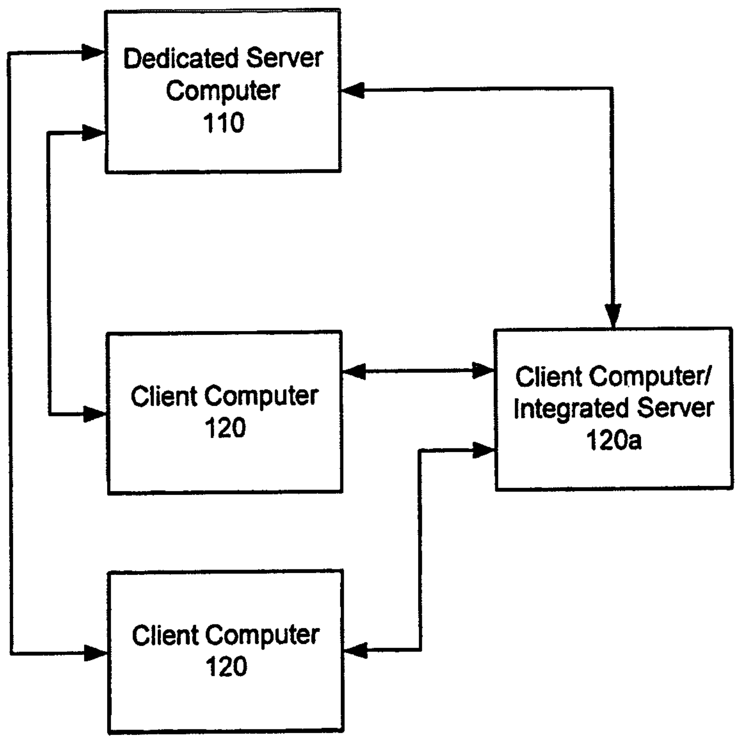

Dedicated Server Computer 110
Client Computer 120
Client Computer/ Integrated Server 120a
Client Computer 120

Figure 2

PRIOR ART

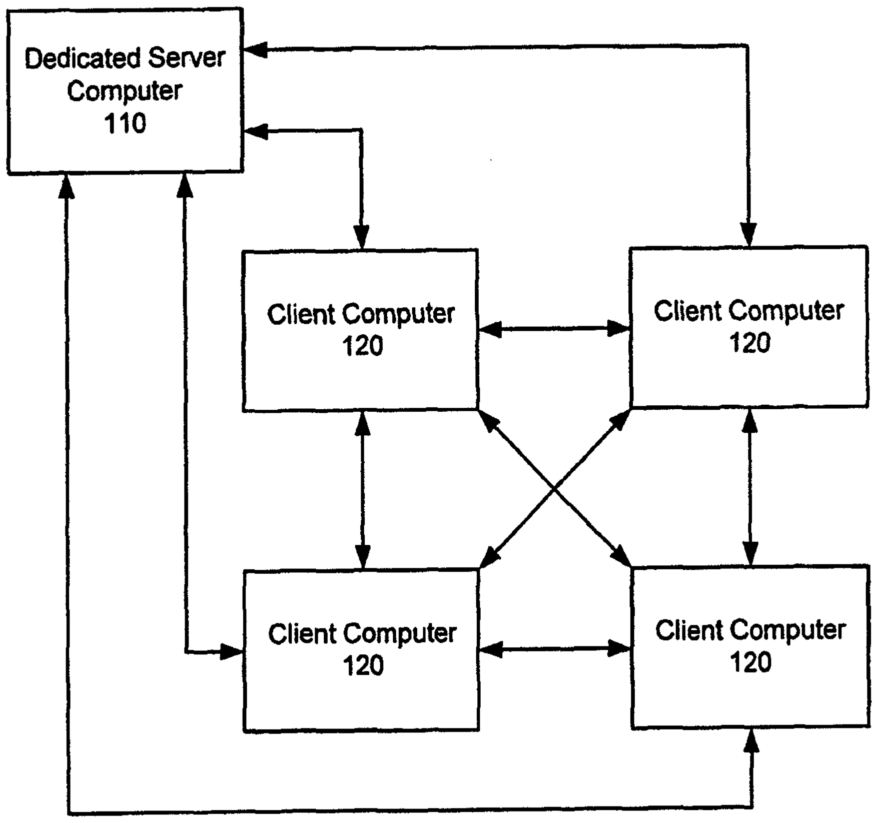


Figure 3

PRIOR ART

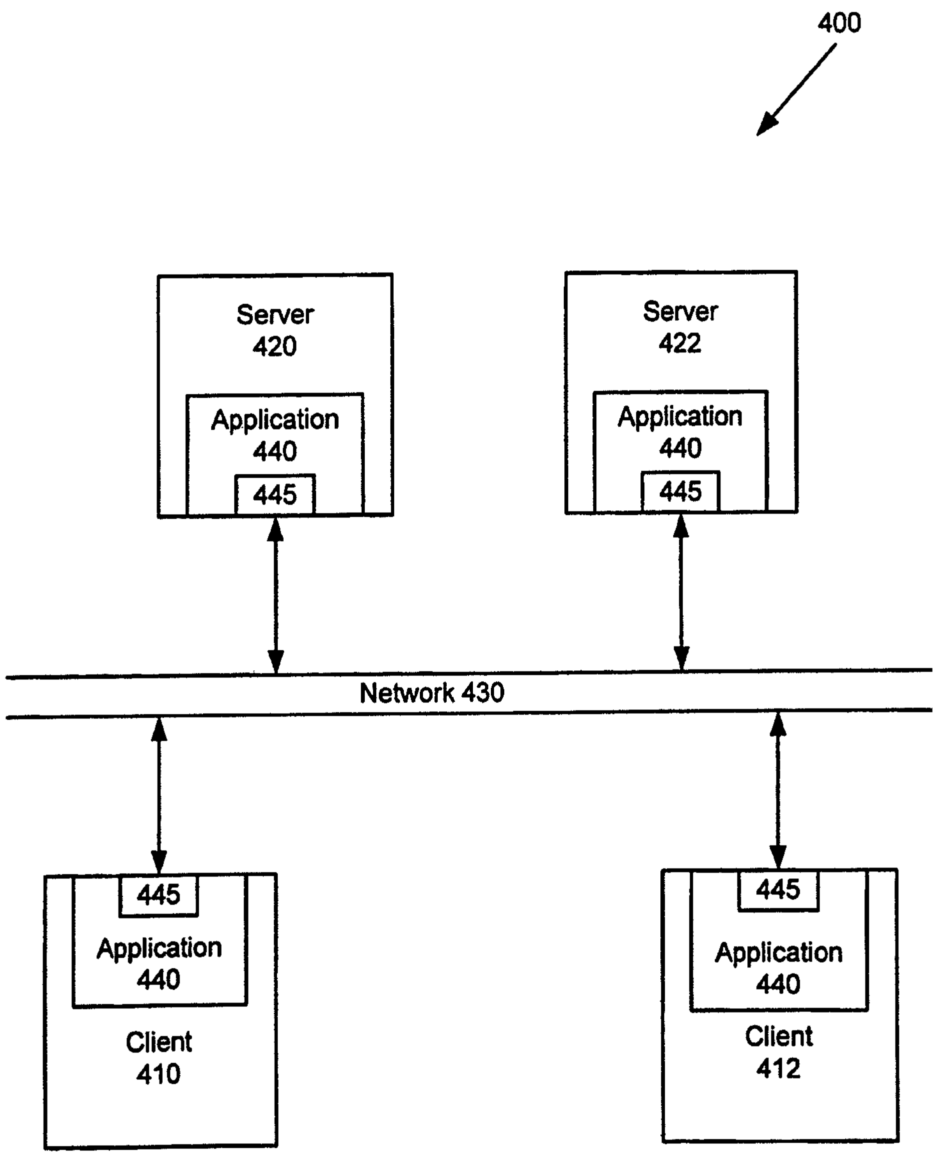

400
Server
420
Application
440
445
Server
422
Application
440
445
Network 430
445
Application
440
Client
410
445
Application
440
Client
412

Figure 4

500

Index	Session Master	Active Port/Protocol Type
1	Owns C1	80/TCP, 90/UDP
2	Owns C2, C3	85/TCP, 95/UDP
3	-	
-	-	
-	-	
-	-	

Figure 5

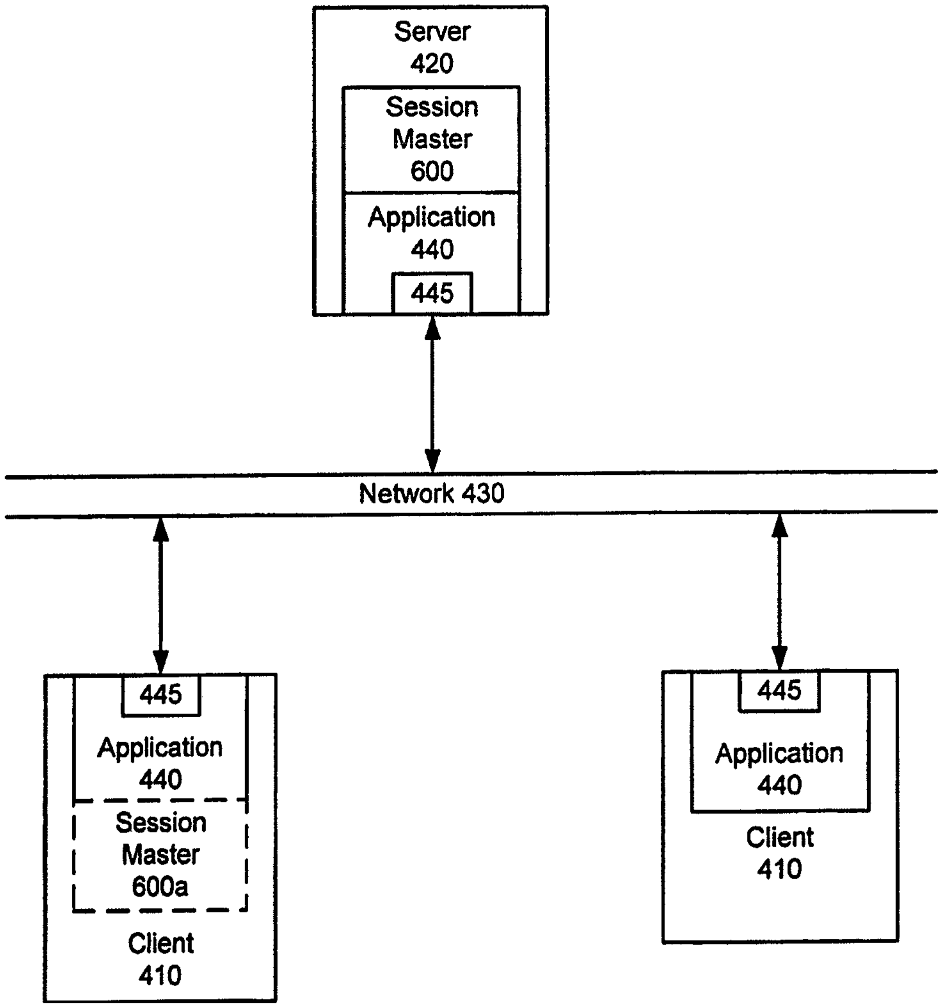

Server
420
Session
Master
600
Application
440
445
Network 430
445
Application
440
Session
Master
600a
Client
410
445
Application
440
Client
410

Figure 6

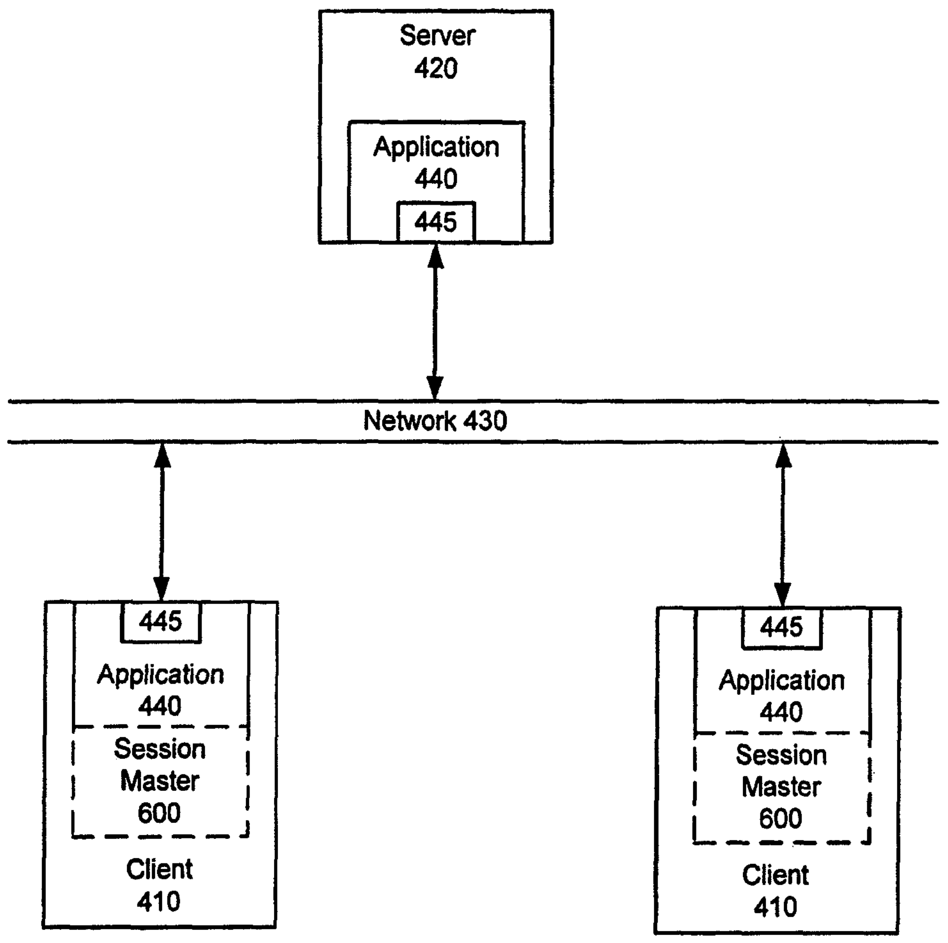

Server
420
Application
440
445
Network 430
445
Application
440
Session
Master
600
Client
410
445
Application
440
Session
Master
600
Client
410

Figure 7

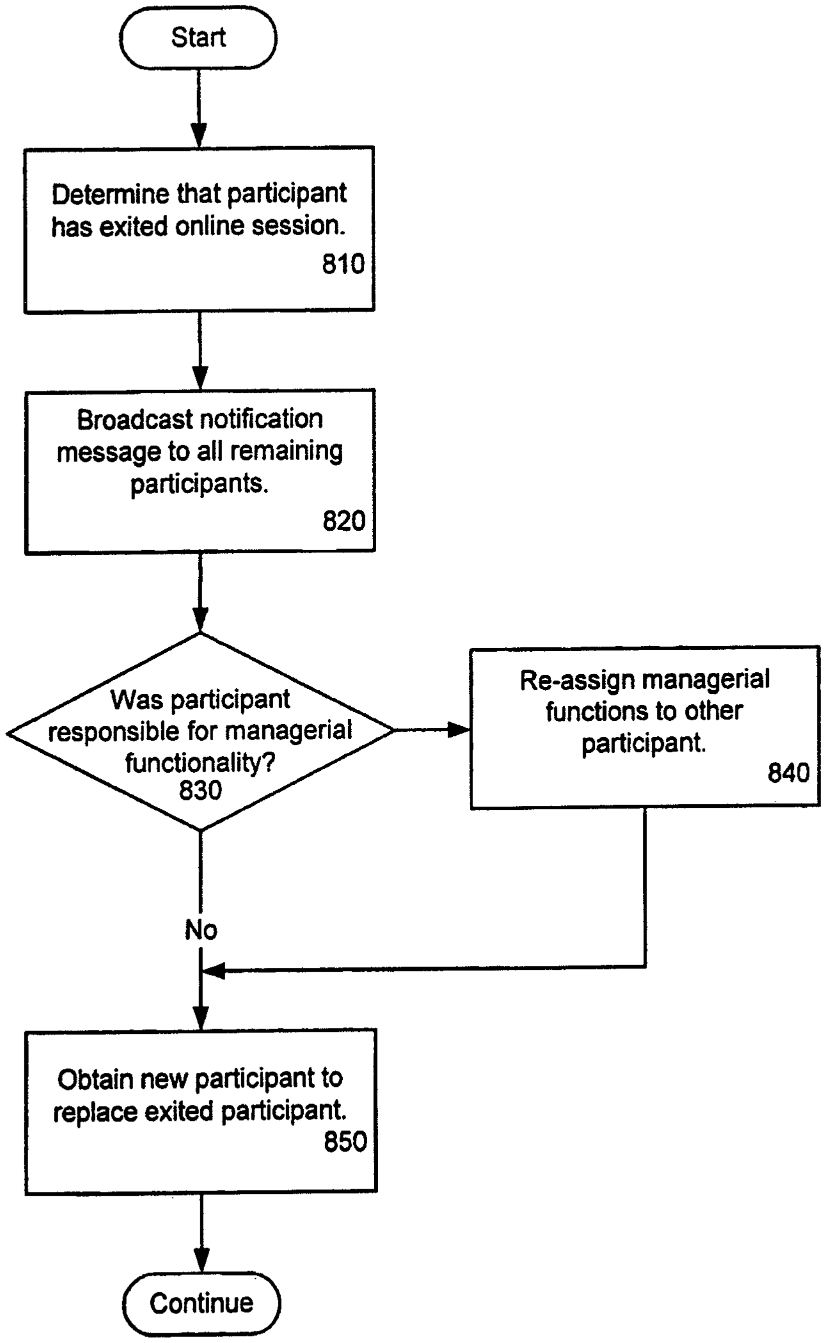

Start
Determine that participant has exited online session. 810
Broadcast notification message to all remaining participants. 820
Was participant responsible for managerial functionality? 830
Re-assign managerial functions to other participant. 840
No
Obtain new participant to replace exited participant. 850
Continue

Figure 8

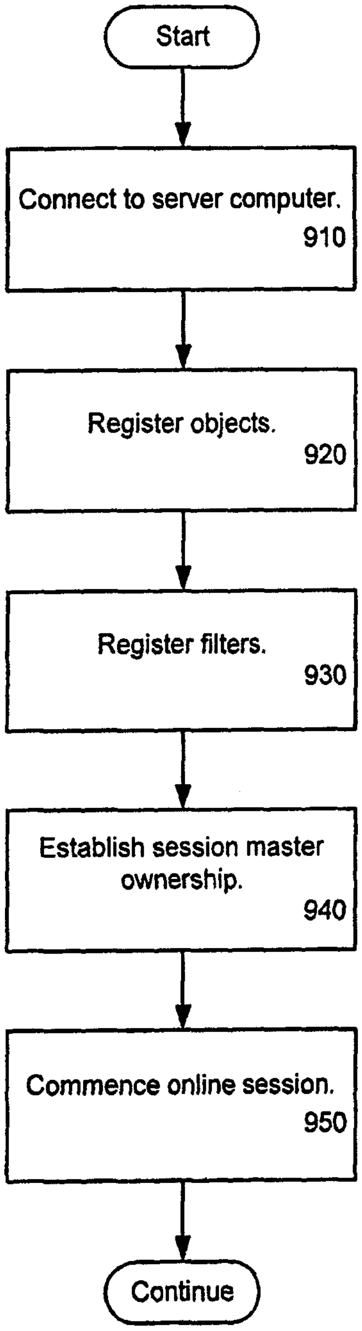

Start
Connect to server computer.
910
Register objects.
920
Register filters.
930
Establish session master ownership.
940
Commence online session.
950
Continue

Figure 9

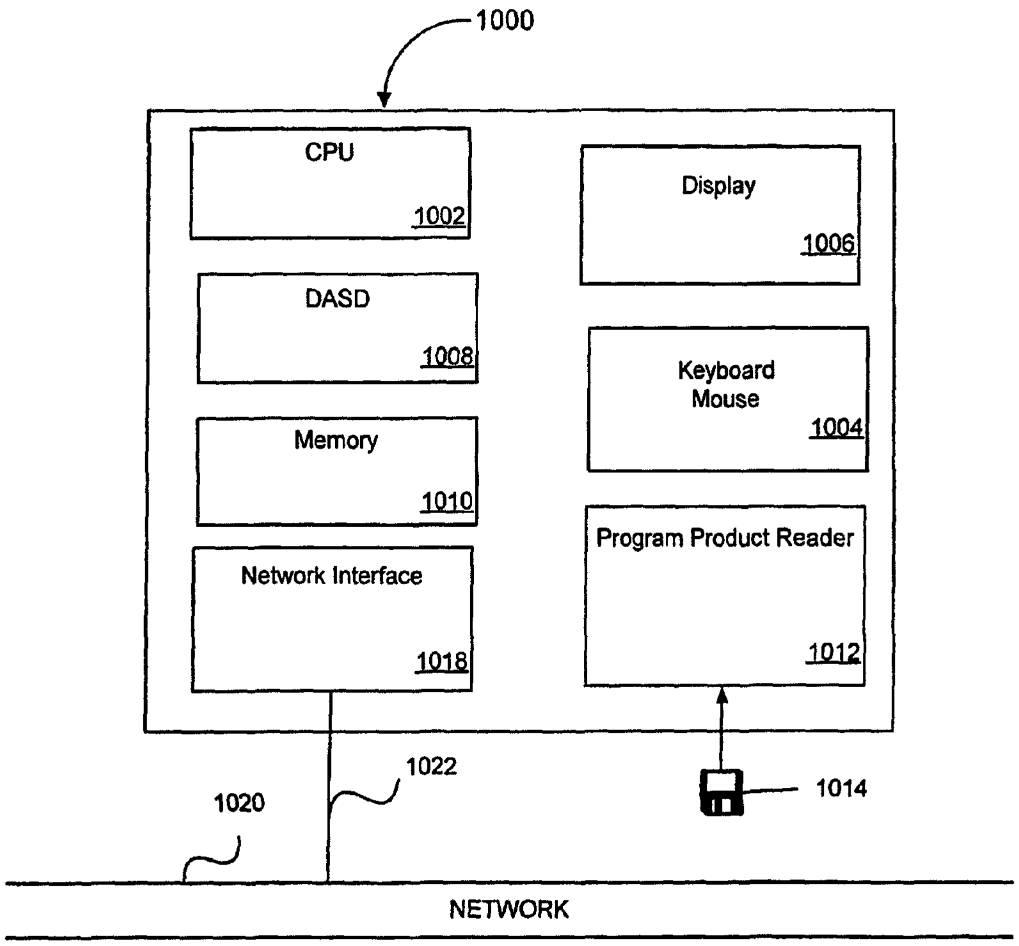

1000
CPU
1002
DASD
1008
Memory
1010
Network Interface
1018
Display
1006
Keyboard
Mouse
1004
Program Product Reader
1012
1022
1020
1014
NETWORK

Figure 10

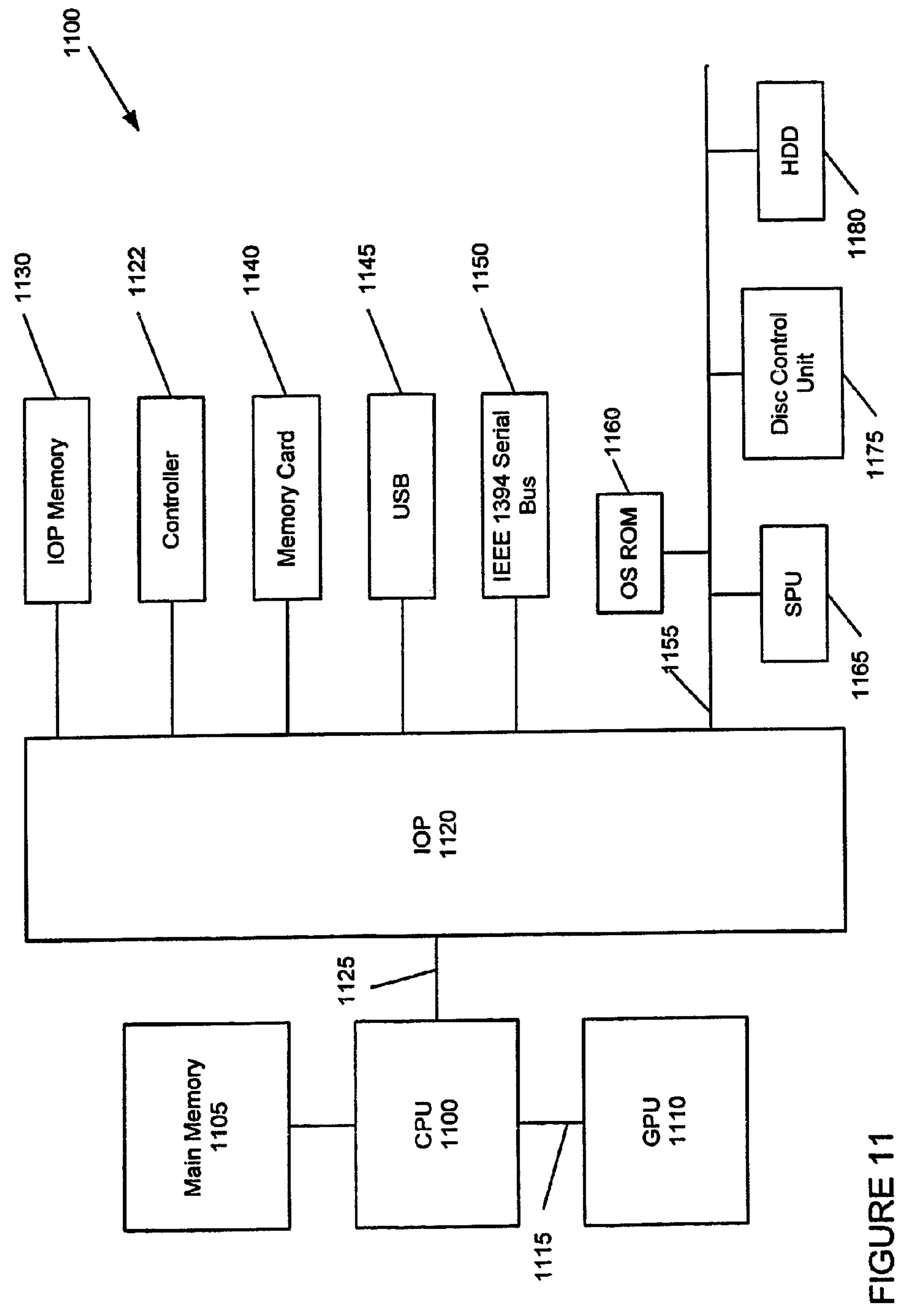

1100
IOP Memory
1130
Controller
1122
Memory Card
1140
USB
1145
IEEE 1394 Serial Bus
1150
OS ROM
1160
1155
HDD
1180
Disc Control Unit
1175
SPU
1165
IOP
1120
1125
Main Memory
1105
CPU
1100
GPU
1110
1115

FIGURE 11

MANAGING PARTICIPANTS IN AN ONLINE SESSION

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation and claims the priority benefit of U.S. patent application Ser. No. 11/375,526 filed Mar. 13, 2006, now U.S. Pat. No. 7,792,902, which is a continuation and claims the priority benefit of U.S. patent application Ser. No. 10/221,128 filed Jul. 31, 2002, now abandoned, which claims the priority benefit of U.S. provisional patent application No. 60/381,736 filed May 17, 2002. The disclosure of these commonly owned applications is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to computer networks and, more particularly, to an application that is run by multiple users on a computer network.

2. Description of the Related Art

Computer networks, such as local area networks and the Internet, are increasingly being used as the backbone for various transactions and interactions between parties. From online banking, where bank customers can initiate financial transactions on a computer network, to online gaming, where garners can participate in various games over the Internet, service providers are increasingly providing a variety of services over computer networks. There are currently a variety of different computer network configurations that facilitate the transactions and interactions that take place.

One type of configuration is a classic client-server configuration, such as is illustrated in FIG. **1**. In this configuration, a dedicated server computer **110** is communicatively linked to one or more client computers **120** over a network, such as through the Internet. The client computer **120** makes service requests to the server computer **110** and the server computer **110** fulfills the request by transmitting data to the requesting client computer **120** over the network. The server computer **110** can be connected to a data storage device or to other computer devices that facilitate transactions between the client and server computers. One characteristic of the client-server configuration is that the client computers cannot communicate directly with one another, as the client computers are limited to communicating with the server computer.

For example, where the client-server configuration is operated in an online gaming environment, the server computer **110** can be responsible for maintaining the various states that are associated with the online game. The server computer can be connected to other computers, such as a memory engine **140** that maintains one or more instances of a game, while the server computer **110** manages administrative matters such as player matching and account management. A game player on the client computer **120** can log onto the server computer **110** and receive a list of available games and participating players. The player chooses a game to start or join, thereby identifying a memory engine with which the player's computer establishes a client-server connection. In this manner, the server computer **110** and the memory engine **140** collectively administer the gaming environment for one or more client computers **120**.

Another type of configuration is referred to as an integrated server configuration, such as is shown in FIG. **2**. This configuration includes a dedicated server computer **110** and one or more client computers **120** that are each connected to the

server computer **110** over a computer network. As in the previously-described configuration, the server computer **110** serves data to the client computers **120**. However, one of the client computers **120**, such as the client computer **120***a*, functions as an integrated server in that the client computer **120***a* can serve data to the other client computers **120**. In an online gaming environment, the server computer **110** can perform administrative functions, such as player matching, account management, and chat room management, while the client computer/integrated server **120***a* can perform the function of the previously-described memory engine.

In yet another type of communication configuration, the various computers are arranged in a peer-to-peer configuration, such as is shown in FIG. **3**. In a peer-to-peer configuration, each of the computers can communicate with the others, so that all of the computers function as "peers." In one form of the peer-to-peer configuration, a dedicated server **110** is communicatively connected to a plurality of client computers **120** over a network. An online session is initially established by each of the client computers **120** connecting to an administrative computer, such as the server computer **110**. The client computers **120** are then communicatively connected to one another so that each of the client computers **120** has the ability to both serve and receive data to and from any of the other client computers **120**. In addition, each client computer **120** can operate in a client-server relationship with the dedicated server **110**. Those skilled in the art will appreciate that there are other communication configurations in addition to the configurations described above.

The various configurations described above enable computer users to interact over a computer network, such as in an online game environment where game players can play computer games on a computer network. In such a scenario, at least one of the computers typically functions as a game manager that manages various aspects of the game, such as coordinating the number of players, keeping track of game state, and sending out updates to the users regarding game state. It can be appreciated that continuity of game play can be highly dependent on all of the users in a game continuing to play throughout the entire game period. Game play can be interrupted or even halted if one of the game players exits during the middle of a game, particularly where the exited player was managing a portion of the game.

For example, sports games typically have a fixed start and a fixed finish for the game, with at least two players competing in a game. In current configurations, there are often several players that participate in an online sporting contest, with each one of the players assuming the role of a player on a sporting team. For example, in an online football game, players can assume the roles of quarterback, receiver, defensive back, running back, etc. If one of the players were to suddenly leave during the middle of the game, then the game play would be interrupted or halted. This could also be the case in other types of games, where continuity of game play is dependent on each of the players in the game environment continuing to play throughout a particular scenario.

Unfortunately, current multi-user applications are not configured to account for when a participant in an online session suddenly or unexpectedly leaves the online session. If a player does leave an online session, then the session is unduly interrupted or terminated. In view of the foregoing, there is a need for a multi-user application that overcomes the aforementioned shortcomings.

SUMMARY OF THE INVENTION

The present invention relates to an application that is configured to be operated in a multi-participant environment on a computer network. The application manages participants in an online session of a multi-user application so that if one of the participants exits the session, the session can continue without interruption. In accordance with one aspect of the invention, the application initiates an online session of the multi-user application, wherein the online session includes two or more participants comprised of network computers that are communicatively linked to a computer network. If the application detects that a first participant has disconnected from the online session, wherein the first participant is responsible for managing certain managerial functionality associated with the running of the multi-user application, then the application broadcasts a notification to existing participants of the online session over the communication network, thereby notifying the existing participants that the first participant has disconnected from the online session. The initiating application then re-assigns the functionality associated with the first participant to an existing participant of the online session. The participants can be communicating in a peer-to-peer arrangement or can be performing server duties in a client-server arrangement.

Other features and advantages of the present invention should be apparent from the following description of the preferred embodiment, which illustrates, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of a computer network arranged in a client-server network communication configuration.

FIG. **2** is an illustration of a computer network arranged in an integrated network communication server configuration.

FIG. **3** is an illustration of a computer network arranged in a peer-to-peer network communication configuration.

FIG. **4** is an illustration of a computer network system on which is run a multi-user application configured in accordance with the present invention.

FIG. **5** is an illustration of a data structure that includes computer index and session master information for the multi-user application.

FIG. **6** is an illustration of a computer network system wherein the multi-user application is arranged in a first type of communication configuration.

FIG. **7** is an illustration of a computer network system wherein the multi-user application is arranged in another type of communication configuration.

FIG. **8** is a flow diagram that represents a process of managing the exiting of a participant in an online session of a multi-user application.

FIG. **9** is a flow diagram that illustrates the operating steps associated with the multi-user application establishing an online session.

FIG. **10** is a block diagram of a computer in the network illustrated in FIG. **4**, illustrating the hardware components.

FIG. **11** is a block diagram of a computer entertainment system in the network illustrated in FIG. **4**, illustrating the hardware components.

DETAILED DESCRIPTION

FIG. **4** is a block diagram of a computer network system **400** comprised of one or more network devices including one or more client computers **410**, **412** and one or more dedicated server computers **420**, **422**, which are nodes of a computer network **430**. Thus, some of the network computers are configured as servers and some are configured as clients. The computer network **430** may comprise a collection of interconnected networks, such as the Internet, and may include one or more local area networks at each of the nodes **410**, **412**, **420**, **422**. As used herein, the term "Internet" refers to a collection of interconnected (public and/or private) networks that are linked together by a set of standard communication protocols to form a global, distributed network.

The client computers **410**, **412** can transmit requests for data over the network **430** to one of the server computers **420**, **422**, which are configured to serve data over the network **430** to the client computers in a well-known manner. The server computers **420**, **422** can include or can be communicatively linked to each other and to other servers, such as a data base server and/or an application server, as will be known to those skilled in the art. Although FIG. **4** shows only two client computers **410**, **412** and two server computers **420**, **422**, it should be appreciated that the network system **400** could include any number of client computers **410**, **412** and server computers **420**, **422**. The server computers **420**, **422** and client computers **410**, **412** are sometimes referred to collectively herein as network computers.

The network system **400** supports a multi-user application **440** comprised of a computer program with which multiple users can interact in online sessions using network devices (such as the client computers **410**, **412**) that are linked to the computer network **430**. The application **440** is installed at each of the client computers, meaning that an operational instance of the application is stored in memory of each of the client computers **410**, **412** that run (execute) the application **440**. Each server computer that will be participating in an online session of the multi-user application also stores an instance of the application **440**. For purposes of this description, the first server computer **420** will be assumed to be a server for the multi-user application being executed by the client machines **410**, **412**, although both servers **420**, **422** are shown with installed applications **440**. An exchange of data occurs between instances of the application **440** during execution and is enabled through the establishment of network sockets **445** at each of the network computers. The sockets are represented in FIG. **4** as boxes at each respective network computer. Those skilled in the art will understand that a network socket is one end of a multi-way communication link between two or more programs that run on the network system **400**.

The application **440** can be run on the network devices of the network system **400** according to a variety of communication configurations and the responsibilities for various application-related processes can be assigned to different computing devices of the network **430**, as described in more detail below. An application development interface is preferably used to develop the application **440**, as is also described in more detail below. The application can be operated such that the associated network computer can use a communications configuration to implement any of the communication modes illustrated in FIG. **1**, FIG. **2**, and FIG. **3**.

The multi-user application **440** can be any type of application that a user can run on a network computer that is linked to the computer network **430**. When the application **440** is run on a client computer **410**, **412**, the user can interact with other users through other network computers that are also running the application **440**. The server computer **420** can function as a central network "meeting point" through which the users can establish contact, maintain data, and initiate an online session of the application **440**. Typically, the application **440** causes the network device in which it is operating to establish communications with another network device, such as the devices **410**, **412**, **420**, thereby initiating an online session.

During the online session, the network computers will interact and exchange data pursuant to the programmed features of the application **440**.

When the application **440** is launched and an online session is established among suitably configured computers, the application enables the computers to interact in a variety of configurations. Throughout this description, the application **440** is sometimes described in an online-gaming scenario, wherein the application **440** comprises a computer game that multiple users can access and run using the client computers **410**, **412**. In such a case, the application **440** establishes an online session comprised of a game in which the network computers participate. However, it should be appreciated that the application **440** may also relate to other scenarios besides gaming, such as, for example, online banking or online travel planning, that involve interactions between multiple computers on a computer network.

When an application **440** executes, it identifies a session master, which is a network computer that performs a variety of managerial and administrative functions for the application with respect to interactions between computers that occur during an online session. An online session of the application uses a registration or logon process with a data store containing information such as user identification. The logon process authorizes further participation in the network environment of the application. Preferably, the session master function is assigned when a client computer running the application **440**, such as the client computer **410**, logs onto the server computer **420** to initiate an online session. The application itself, however, determines the details of when and how such assignments are made, so that a variety of session master assignment schemes can be implemented without departing from the teachings of the present invention.

The operating instance of the application on the client computer **410** that initiates an online session of the application is referred to as the host computer. The application at the host computer assigns the session master function to either the server computer **420** or to the host computer **410**. As new client computers log-on (register) with the server computer **420** to join the online session, the server computer **420** notifies the new clients of the already-assigned identity of the session master computer.

As described more fully below, the session master functionality enables a smooth transition between the various network communication configurations in which the application **440** can operate. The session master function also enables the application **440** to concentrate responsibility for application-related tasks in a specific network computer, or to distribute such responsibility among two or more network computers. The assignment of tasks can be performed by an instance of the application **440** on one of the network computers, at the same time when the session master function is assigned, and the session master tasks can be assigned to one or more of the computers on the network **430** for providing the requisite functionality. The computer or computers that are assigned responsibility of the session master are referred to herein as the "owners" of the respective session master functions. References to a solitary session master will be understood to apply to a group of computers, if those computers are collectively performing the session master functions. Thus, the assignment of session master tasks is performed in the manner specified by the application, in accordance with the dictates of the application developer.

One category of responsibilities assigned to the session master relates to application-specific functions, which are functions that are peculiar to the particular type of application **440** being executed. For example, if the application **440** is a

game-type application, then the session master or a group of session masters can keep track of game-type data, such as the game score and the time remaining in the game, and can perform game functions, such as terminating the online session when a game ends. A session master computer can also be assigned the responsibility of keeping track of specific game data, such as the state of an object in the game environment, such as a football, aircraft, ocean, tree, and so forth. Each of these responsibilities can be concentrated in a single session master computer or can be divided up among several session master computers, in accordance with the operation of the application.

The host computer performs managerial functions related to the computers that are participating in the online session. For example, whenever a network computer joins an online session of the application **440**, the host computer assigns an identification index number to the computer joining the session. The host computer maintains a list of the identification index numbers and their associated network computers. The index number is used when sending messages and is also used to maintain ownership records regarding the session master functionality.

As noted above, there can be more than one session master in an online session. The way in which a session master is assigned can be determined by the application in accordance with the operation of the application. The application **440** can also assign the session master with responsibility for sending out update messages to update the network computers regarding the status of all network computers that are participating in the online session. This responsibility entails the session master notifying the participating network computers when a new network computer joins the online session, or when a current participant exits the online session of the application **440**, as described more fully below.

The host computer that assigns the aforementioned index number to each of the computers also maintains a list of all network computers that are participating in the online session. The application **440** then keeps track of session master ownership according to the index number assigned to the computer. To keep track of the index number and responsibility assignments, the application **440** can maintain a data structure such as in the form of a table comprised of a network computer index list, such as the table **500** shown in FIG. **5**. The table **500** contains an index number associated with each network computer that is participating in the online session, and also contains an indication of whether the network computer owns the session master function. The index list data structure comprising the table **500** preferably also specifies the communication protocol that is being used for each network computer. FIG. **5** shows that different session master tasks (C**1**, C**2**, C**3**) can be owned by different network computers.

In addition to specifying the communication protocol, the data structure also specifies, for each network computer, the port for which the communication protocol is associated. Each instance of the application **440** enables the associated network computer that is participating in the online session to open multiple communication ports, with each port being associated with a particular communication protocol. The network computers communicate with other network computers using a particular port and a particular protocol, which is specified in the data structure comprised of the table **500** shown in FIG. **5**. The ports can comprise network sockets through which the instances of the application **440** communicate over the network. The network computers preferably communicate the port/protocol information, as well as the other information contained in the index list, by periodically sending communication messages to one another over the network.

Preferably, all of the computers that are participating in the online session keep their own copy of the table **500** index list. It should be appreciated that the table **500** is merely exemplary and that the initiating host application **440** could keep track of client index numbers and session master ownership in other manners using a wide variety of data structure formats. Alternatively, the session computers can share one or more copies of the table.

The initiating application **440** can operate in various communication configurations depending on how the application **440** assigns ownership of the session master. In a first configuration, shown in FIG. **6**, the initiating application **440** has assigned ownership of a session master **600** to a single computer, such as the dedicated server computer **420**. Therefore, this computer **420** has responsibility for all of the tasks associated with the session master function as dictated by the initiating application **440**. Thus, the application **440** operates in a client-server communication configuration with respect to the functions of the session master, with the server computer **420** serving data to the client computers **410** relating to the session master responsibilities.

It should be appreciated that any of the computers that are participating in the online session of the application **440** could have ownership of one or more of the session master tasks, such as where one of the client computers **410** is shown owning a session master **600***a*, which is shown in FIG. **6** using phantom lines. This indicates that the client computer has been assigned and is performing one or more session master tasks, along with or in place of the nominal session master server computer **420**. That is, there can be several instances of a session master among the computers participating in an online session, with each instance of a session master being assigned specific responsibilities and each session master task being assigned to a different network computer, or multiple tasks being assigned to the same computer. For example, FIG. **6** shows a situation where there are two session masters **600** and **600***a*, each being assigned responsibility for certain functionality relating to the online session of the application **440**. The server computer **420** has certain responsibilities and the client computer **410** also has certain responsibilities, as determined by the application. This is an integrated server configuration, where the client computer **410** that owns the session master **600***a* is functioning as an integrated server. An "integrated server" refers to a situation in which all clients send information to a client that is designated as an integrated server, and where that integrated server propagates the information to the other clients. The client acting as the integrated server can also own one or more session master tasks.

In another scheme, shown in FIG. **7**, the application **440** has distributed ownership of the session master **600** among several computers. In the illustrated example, both of the client computers **410** share ownership of the session master **600**. In such a case, both of the computers could perform the functions associated with the session master so that the network computers in FIG. **7** are in a peer-to-peer configuration. Regardless of the particular communication configuration in which the application **440** operates, the online session of the application includes various network participants comprised of the network computers that are running the application and interacting pursuant to the online session. If one of the participants in the online session were to exit the session, then this may result in an interruption for the other participants that remain in the session. In accordance with one aspect of the invention, the application **440** is configured to handle situations where a participant exits an online session in order to minimize the interruption for remaining participants.

This is described in more detail with reference to the flow diagram shown in FIG. **8**, which describes a process of managing the exiting of a participant in an online session of the multi-user application **440**. In the first operation, represented by the flow diagram box numbered **810**, it is determined that a participant of the online session of the application has exited the online session. The determination that a participant has exited the online session can be made in a variety of manners. In one embodiment, the instance of the application **440** on a network computer participating in the online session periodically causes the network computer to broadcast an update message notifying the other network computers of its presence in the online session. If no update message is received from a particular network computer within a predetermined amount of time, then it is deemed that the network computer has exited the online session. During establishment of the online session, the instance of the application **440** on the computer that starts the online session may assign a particular computer, such as the session master computer, with the responsibility for making the determination that a participant has exited the online session.

In the next operation, represented by the flow diagram box numbered **820**, the instance of the application **440** in one of the computers causes a notification message to be broadcast to all of the participants in the online session notifying them that a participant (the "exited participant") has exited the session. Only one of the network computers, such as, for example, the session master computer, broadcasts the notification message to all of the participants. If the session master computer is the exited computer, then one of the other network computers broadcasts the message, such as the computer with the next consecutive index after the session master computer. The notification message preferably includes the index that was previously assigned to the network computer for the exited participant. In this manner, the other participants can identify the exited participant by consulting the index table that was discussed above with respect to FIG. **5**.

The next operation differs based upon whether the participant that exited the session was responsible for performing any session managerial functions that would affect the other participants of the online session, as represented by the decision box numbered **830**. The managerial functions include functions such as filtering communication messages, assigning identification indices, score-keeping, keeping track of session times, keeping track of items that are located in an online world, keeping track of participant locations in an online world, etc. If the participant that exited the session was responsible for performing any such functions, a "Yes" result from the decision box **830**, then the process proceeds to the operation represented by the flow diagram box numbered **840**. In this operation, the application **440** reassigns the managerial responsibilities of the exited participant to a network computer of another participant that is still present in the online session. The reassignment of managerial responsibilities is preferably performed by an instance of the application **440** on a specific computer, such as the computer that has the next consecutive index after the index of the computer that exited the online session. For example, a computer with a first index may exit the session. The instance of the application on computer with the next consecutive index (as specified in the table **500** shown in FIG. **5**) then performs the re-assignment of the exited computer's duties.

The manner in which the application **440** re-assigns the responsibilities can vary. In one embodiment, the application **440** automatically selects the participant that is re-assigned the responsibilities based on certain factors, some of which are related to the conditions of the network computers of the participants. The conditions can include, for example, the communication environment, geographical location, and hardware specification of the network computers, as well as user-specified preferences.

The communication environment relates to whether the network computer of the participant has large bandwidth capabilities, such as through a cable-modem or DSL. Preferably, those participants with higher bandwidth capabilities are given higher preference for assuming the responsibilities of the exited participant. The geographic location of the participants can also be a factor in deciding which participant is re-assigned the responsibilities of the exited participant. For example, a participant that is centrally located to the other participants may be given a higher priority in order to minimize the latency for communications. The hardware specifications of the network computers that are participating in the online session are also a factor. The application **440** may give higher priority to network computers that have hardware capabilities that are most highly suited for the responsibilities that are being re-assigned, such as computers with powerful data processing capabilities.

In another embodiment of the operations of flow diagram box **840**, the application **440** simply randomly re-assigns the responsibilities of the exited participant to another of the participants of the online session. The application **440** can also consider user-specified preferences. Some users may specify to the application **440** that they do not want to be assigned the responsibility of managing any application functionality. The users may also specify that a particular participant should be re-assigned responsibilities should another participant exit the online session. Alternately, the application **440** may cause a message to be broadcast to all of the participants of the online session asking whether any of the participants would like to take over the responsibilities that were previously assigned to the exited participant.

The next operation is represented by the flow diagram box numbered **850**. This operation occurs after the application **440** has re-assigned the responsibilities of the exited participant. The operation of flow diagram box **850** also occurs if the exited participant did not have any responsibilities that needed re-assignment, which would have resulted in a "No" outcome from the decision box numbered **830**. In this operation, the application **440** attempts to obtain a new participant to replace the exited participant. The attempt is preferably performed by the instance of the application **440** on a specific computer, such as the session master computer. It should be appreciated that this operation differs from the operation of flow diagram box **840** in that this operation relates to obtaining a replacement participant for the online session to take the place of the exited participant, rather than re-assigning the managerial functions of the exited participant.

For example, the online session may be an online football game, where the participants are each a player on a common team. One of the participants may have been assigned managerial functions comprised of keeping track of the score and of the game time. That same participant may have played the role of the quarterback in the game. If the participant exits the online game during the game, then the application **440** re-assigns the managerial functions (i.e., score keeping and game time responsibilities) of the exited participant to another participant in the operation **840** and then, in operation **850**, attempts to obtain a new participant to replace the exited participant's role as quarterback.

The manner in which the application **440** attempts to obtain a replacement participant for the online session may vary. In one embodiment, the application **440** automatically assigns a network computer, rather than a human, as a replacement for the exited participant. The network computer thereby would perform the functions of the exited participant. In another embodiment, the application **440** maintains a list of network computer that might be able to participate in the online session and then sends a message to those computers inviting them to participate in the session. The application **440** may put the online session in a pause mode while a replacement participant is obtained.

The application **440** is preferably developed using a software development kit (SDK) that provides a library of object and communication message definitions that are used in the application **440**. The software development kit includes an application interface through which applications that are developed using the SDK can run on a network system, such as the network system **400**. The application interface can reside in a central network server, such as the server **420**, to which network computers that have the application **440** can log onto in order to operate an online session of the application. By using the object and message types provided by the SDK, the application **440** can be developed to include the features described above. The SDK preferably includes an object definition structure that provides a client-based definition of objects that are utilized by the application **440**. The object definition includes a plurality of characteristics associated with each object and utilized by the application to effect interaction with clients over the computer network.

Once the application **440** has been developed using the SDK, the application **440** can be loaded onto one or more network computers and an online session can be established according to the operations shown in the flow diagram box of FIG. **9**. In the first operation, represented by the flow diagram box numbered **910**, a network computer on which the application **440** is loaded connects to a network computer that includes in memory that application interface software. For example, one or more of the client computers **410** of the network system **400** shown in FIG. **4** may have the application **440** loaded in memory and the server computer **420** may include the application interface. In such a case, the client computers **410** establish a communication connection with the server computer **410** over the network **430**.

In the next operation, represented by the flow diagram box numbered **920**, the application **440** registers objects according to the object definitions that are available in the library of the application interface. The application **440** also registers any message filters that will be utilized during the online session, as represented by the flow diagram box numbered **930**.

In the next operation, represented by the flow diagram box numbered **940**, the application **440** defines the session master and assigns ownership of the session master to one of the network computers. The ownership of the session master can be assigned to one computer or can be assigned to plural computers. The application **440** also specifies whether the ownership of the session master is dedicated to a particular computer or whether ownership can migrate to other computers.

During this operation, the application **440** assigns client indices to each of the network computers that will participate in the online session and also establishes the index table described above. The application **440** can be configured such that the first network computer to log onto the server computer will be the session master and also receive an initial index, such as an index of one or zero. It should be appreciated that the initial index may vary. Subsequent network computers to log on will then receive the next available index. After the ownership of the session master or session masters has been established, the online session of the application **440** is commenced, as represented by the flow diagram box numbered **950**.

As noted above, the network computers shown in the block diagram of FIG. **4** comprise nodes of a computer network system **400**. FIG. **10** is a block diagram of a computer in the system **400** of FIG. **4**, illustrating the hardware components included in one of the computers. Those skilled in the art will appreciate that the devices **410** and **420** may all have a similar computer construction, or may have alternative constructions consistent with the capabilities described herein.

FIG. **10** shows an exemplary computer **1000** such as might comprise any of the network computers. Each computer **1000** operates under control of a central processor unit (CPU) **1002**, such as a "Pentium" microprocessor and associated integrated circuit chips, available from Intel Corporation of Santa Clara, Calif., USA. A computer user can input commands and data from a keyboard and computer mouse **1004**, and can view inputs and computer output at a display **1006**. The display is typically a video monitor or flat panel display. The computer **1000** also includes a direct access storage device (DASD) **1008**, such as a hard disk drive. The memory **1010** typically comprises volatile semiconductor random access memory (RAM). Each computer preferably includes a program product reader **1012** that accepts a program product storage device **1014**, from which the program product reader can read data (and to which it can optionally write data). The program product reader can comprise, for example, a disk drive, and the program product storage device can comprise removable storage media such as a magnetic floppy disk, a CD-R disc, a CD-RW disc, or DVD disc.

Each computer **1000** can communicate with the others over a computer network **1020** (such as the Internet or an intranet) through a network interface **1018** that enables communication over a connection **1022** between the network **1020** and the computer. The network interface **1018** typically comprises, for example, a Network Interface Card (NIC) or a modem that permits communications over a variety of networks.

The CPU **1002** operates under control of programming steps that are temporarily stored in the memory **1010** of the computer **1000**. When the programming steps are executed, the computer performs its functions. Thus, the programming steps implement the functionality of the application **440**. The programming steps can be received from the DASD **1008**, through the program product storage device **1014**, or through the network connection **1022**. The program product storage drive **1012** can receive a program product **1014**, read programming steps recorded thereon, and transfer the programming steps into the memory **1010** for execution by the CPU **1002**. As noted above, the program product storage device can comprise any one of multiple removable media having recorded computer-readable instructions, including magnetic floppy disks and CD-ROM storage discs. Other suitable program product storage devices can include magnetic tape and semiconductor memory chips. In this way, the processing steps necessary for operation in accordance with the invention can be embodied on a program product.

Alternatively, the program steps can be received into the operating memory **1010** over the network **1020**. In the network method, the computer receives data including program steps into the memory **1010** through the network interface **1018** after network communication has been established over the network connection **1022** by well-known methods that will be understood by those skilled in the art without further explanation. The program steps are then executed by the CPU **1002** thereby comprising a computer process.

It should be understood that all of the network computers of the network system **400** illustrated in FIG. **4** may have a construction similar to that shown in FIG. **10**, so that details described with respect to the FIG. **10** computer **1000** will be understood to apply to all computers of the system **400**. It should be appreciated that any of the network computers can have an alternative construction, so long as the computer can communicate with the other computers over a network as illustrated in FIG. **4** and can support the functionality described herein.

For example, with reference to FIG. **11**, the client computers **420** can comprise a computer entertainment system, such as a video game system **1100**. FIG. **11** is a block diagram of an exemplary hardware configuration of the video game system **1100**.

The video game system **1100** includes a central processing unit (CPU) **1100** that is associated with a main memory **1105**. The CPU **1100** operates under control of programming steps that are stored in the OS-ROM **1160** or transferred from a game program storage medium to the main memory **1105**. The CPU **1100** is configured to process information and to execute instructions in accordance with the programming steps.

The CPU **1100** is communicatively coupled to an input/output processor (IOP) **1120** via a dedicated bus **1125**. The IOP **1120** couples the CPU **1100** to an OS ROM **1160** comprised of a non-volatile memory that stores program instructions, such as an operating system. The instructions are preferably transferred to the CPU via the IOP **1120** at start-up of the main unit **1100**.

The CPU **1100** is communicatively coupled to a graphics processing unit (GPU) **1110** via a dedicated bus **1115**. The GPU **1110** is a drawing processor that is configured to perform drawing processes and formulate images in accordance with instructions received from the CPU **1100**. For example, the GPU **1110** may render a graphics image based on display lists that are generated by and received from the CPU **1100**. The GPU may include a buffer for storing graphics data. The GPU **1110** outputs images to an audio-visual output device.

The IOP **1120** controls the exchange of data among the CPU **1100** and a plurality of peripheral components in accordance with instructions that are stored in an IOP memory **1130**. The peripheral components may include one or more input controllers **1122**, a memory card **1140**, a USB **1145**, and an IEEE 1394 serial bus **1150**. Additionally, a bus **1155** is communicatively coupled to the IOP **1120**. The bus **1155** is linked to several additional components, including the OS ROM **1160**, a sound processor unit (SPU) **1165**, an optical disc control unit **1175**, and a hard disk drive (HDD) **1180**.

The SPU **1165** is configured to generate sounds, such as music, sound effects, and voices, in accordance with commands received from the CPU **1100** and the IOP **1120**. The SPU **1165** may include a sound buffer in which waveform data is stored. The SPU **1165** generates sound signals and transmits the signals to speakers.

The disc control unit **1175** is configured to control a program reader, which can comprise, for example, an optical disk drive that accepts removable storage media such as a magnetic floppy disk, an optical CD-ROM disc, a CD-R disc, a CD-RW disc, a DVD disk, or the like.

The memory card **1140** may comprise a storage medium to which the CPU **1100** may write and store data. Preferably, the memory card **1140** can be inserted and removed from the IOP **1120**. A user can store or save data using the memory card **1140**. In addition, the video game system **1100** is preferably provided with at least one hard disk drive (HDD) **1180** to which data may be written and stored.

A data I/O interface, such as an IEEE 1394 serial bus **1150** or a universal serial bus (USB) **1145** interface, is preferably communicatively coupled to the IOP **1120** in order to allow data to be transferred into and out of the video game system **1100**, such as to the network **430** of FIG. **4**.

The system and method described above improve on the situation where a network user of an application, such as a game player, is performing as an Integrated Server (IS) for the application, thereby maintaining an application environment, so that the application would end for a conventional implementation of the application when that IS user wishes to log off. As described above, some applications (such as multi-user gaming applications) alternatively permit the functions (and data) of the departing user to be migrated from the departing user to a different user, who will continue with the online session and will take over the duties of the IS. This type of hand-off is typically rather cumbersome and might not be accomplished smoothly. In the case of a gaming environment, for example, a departing player might abruptly disappear from the game environment, thereby disrupting the gaming experience of the other players. The multi-user application in accordance with the invention permits continued use of the application, even with users departing and joining, by notifying all user machines when another user has departed from the session. Suitable adjustments can be made for a more pleasing application environment. The notification occurs through a Disconnect function that ensures proper IS operation and communications. That is, an application server or IS of the application can broadcast a message to all clients in an application environment to notify them that a user has departed or has joined, and can ensure appropriate functionality, if necessary, of the users.

For a system with a network device operating as an Integrated Server (IS) as described above by serving application data to other users, the failure or departure of an IS from the application environment is responded to by assigning a different user as a new IS. The application can assign a new IS by automatically carrying out a replacement process, or by sending a broadcast message to all users and awaiting replies. For automatic selection, the application can assign the new IS in accordance with considerations that include the bandwidth available to the user, the geographic location of the potential new IS, a user's indicated preference for consideration as an IS, the technical specification and resources available at the user's machine, or through a random selection process. If the application is designed so that it sends a broadcast message, then the message will typically solicit voluntary agreement to operate as the new IS.

In addition, a new user who logs in to the system after an IS failure and who otherwise might have been registered with the failed IS can instead be diverted to a different IS, reducing the workload of the group being served by the now-unavailable IS. In this way, newcomers who wish to join the application environment of an IS can instead be moved to a different IS and different user group. Alternatively, the application can respond to a failed IS by dissolving or disbanding the online session of the group administered by the unavailable IS and forming a new online group with a new IS. These alternatives can be selected by an application developer who is configuring an application for operation in accordance with the present invention.

If an individual user leaves during an online session, the result can be somewhat more problematic. In the online gaming context, for example, a certain minimum number of users (players) are required for a game to proceed. In accordance

with the invention, the application can respond by sending a message to other users on the network, inviting others to join the online session and participate in the multi-user application (such as a game). Alternatively, the application can be configured so as to invoke an Artificial Intelligence module to carry out the duties of the Integrated Server.

The present invention has been described above in terms of a presently preferred embodiment so that an understanding of the present invention can be conveyed. There are, however, many configurations for the system and application not specifically described herein but with which the present invention is applicable. The present invention should therefore not be seen as limited to the particular embodiment described herein, but rather, it should be understood that the present invention has wide applicability with respect to multi-user applications generally. All modifications, variations, or equivalent arrangements and implementations that are within the scope of the attached claims should therefore be considered within the scope of the invention.

What is claimed is:

1. A method of managing participants in an online session, the method comprising:

detecting that a first participant has disconnected from the online session including three or more communicatively coupled participant computing devices, wherein the first participant is responsible for managerial functionality associated with the online session, and wherein the detection of the disconnected first participant is performed by one of the remaining participants of the online session;

broadcasting a notification to the remaining participants of the online session, the notification broadcast by one of the remaining participants of the online session following detection that the first participant has disconnected from the online session, the notification indicating that the first participant has disconnected from the online session;

re-assigning the functionality associated with the first participant to one of the remaining participants of the online session, wherein the re-assignment is based on user-specified preferences of the remaining participants and wherein the user-specified preferences comprise willingness or unwillingness to be re-assigned the functionality; and

obtaining a new participant after the first participant has disconnected from the online session to replace the first participant, wherein the new participant is not currently a part of the online session and the new participant replaces the first participant without prior approval or an election involving the first participant or any of the remaining participants.

2. The method of claim **1**, wherein the new participant is a computing device controlled by a human user.

3. The method of claim **1**, wherein the new participant is a computing device not controlled by a human user.

4. A method of managing participants in an online session, the method comprising:

detecting that a first participant has disconnected from the online session including three or more communicatively coupled participant computing devices, wherein the first participant is responsible for managing certain functionality associated with the online session, and wherein the detection of the disconnected first participant is performed by one of the remaining participants of the online session;

broadcasting a notification to the remaining participants of the online session, the notification broadcast by one of the remaining participants of the online session following detection that the first participant has disconnected from the online session, the notification indicating that the first participant has disconnected from the online session;

obtaining a new participant after the first participant has disconnected from the online session to replace the first participant, wherein the new participant is not currently a part of the online session and the new participant replaces the first participant without prior approval or an election involving the first participant or any of the remaining participants; and

re-assigning the functionality associated with the first participant to the newly obtained participant, wherein the re-assignment is based on user-specified preferences of the newly obtained participant and wherein the user-specified preferences comprise willingness or unwillingness to be re-assigned the functionality.

5. The method of claim **4**, wherein the new participant is a computing device controlled by a human user.

6. The method of claim **4**, wherein the new participant is a computing device not controlled by a human user.

7. A method of managing participants in an online session, the method comprising:

detecting that a first participant has disconnected from the online session including three or more communicatively coupled participant computing devices, wherein the first participant is responsible for managing certain functionality associated with the online session, and wherein the detection of the disconnected first participant is performed by one of the remaining participants of the online session;

broadcasting a notification to the remaining participants of the online session, the notification broadcast by one of the remaining participants of the online session following detection that the first participant has disconnected from the online session, the notification indicating that the first participant has disconnected from the online session;

re-assigning the functionality associated with the first participant to one of the remaining participants of the online session, wherein the reassignment is based on:

user-specified preferences of the remaining participants and wherein the user-specified preferences comprise willingness or unwillingness to be re-assigned the functionality, and

bandwidth capability of a communications network associated with one of the remaining participants, wherein the bandwidth capability of the communications network is higher than a communications network associated with any of the remaining participants; and

obtaining a new participant after the first participant has disconnected from the online session to replace the first participant, wherein the new participant is not currently a part of the online session and the new participant replaces the first participant without prior approval or an election involving the first participant or any of the remaining participants.

8. A method of managing participants in an online session, the method comprising:

detecting that a first participant has disconnected from the online session including three or more communicatively coupled participant computing devices, wherein the first

participant is responsible for managing certain functionality associated with the online session, and wherein the detection of the disconnected first participant is performed by one of the remaining participants of the online session;

broadcasting a notification to the remaining participants of the online session, the notification broadcast by one of the remaining participants of the online session following detection that the first participant has disconnected from the online session, the notification indicating that the first participant has disconnected from the online session;

re-assigning the functionality associated with the first participant to one of the remaining participants of the online session, wherein the reassignment is based on

user-specified preferences of the remaining participants and wherein the user-specified preferences comprise willingness or unwillingness to be re-assigned the functionality, and

randomness among the remaining participants willing to be re-assigned the functionality; and

obtaining a new participant after the first participant has disconnected from the online session to replace the first participant, wherein the new participant is not currently a part of the online session and the new participant replaces the first participant without prior approval or an election involving the first participant or any of the remaining participants.

9. A method of managing participants in an online session, the method comprising:

detecting that a first participant has disconnected from the online session including three or more communicatively coupled participant computing devices, wherein the first participant is responsible for managing certain functionality associated with the online session, and wherein the detection of the disconnected first participant is performed by one of the remaining participants of the online session;

broadcasting a notification to the remaining participants of the online session, the notification broadcast by one of the remaining participants of the online session following detection that the first participant has disconnected from the online session, the notification indicating that the first participant has disconnected from the online session;

re-assigning the functionality associated with the first participant to one of the remaining participants of the online session, wherein the reassignment is based on:

user-specified preferences of the remaining participants and wherein the user-specified preferences comprise willingness or unwillingness to be re-assigned the functionality, and

a predetermined order; and

obtaining a new participant after the first participant has disconnected from the online session to replace the first participant, wherein the new participant is not currently a part of the online session and the new participant replaces the first participant without prior approval or an election involving the first participant or any of the remaining participants.

* * * * *

EXHIBIT 95

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

	§	
UNILOC USA, INC. and UNILOC LUXEMBOURG S.A.,	§	
	§	CIVIL ACTION NO. 6:13-cv-259
Plaintiffs,	§	
	§	**JURY TRIAL DEMANDED**
v.	§	
	§	
ELECTRONIC ARTS, INC.,	§	
	§	
Defendant.	§	
	§	

PLAINTIFFS' ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT

Plaintiffs Uniloc USA, Inc. ("Uniloc USA") and Uniloc Luxembourg S.A. ("Uniloc Luxembourg") (collectively, "Uniloc") file this Original Complaint against Defendant Electronic Arts, Inc. for infringement of U.S. Patent No. 5,490,216 ("the '216 patent").

THE PARTIES

1. Uniloc USA, Inc. ("Uniloc USA") is a Texas corporation with its principal place of business at Legacy Town Center I, Suite 380, 7160 Dallas Parkway, Plano, Texas 75024. Uniloc USA also maintains a place of business at 102 N. College, Ste. 806, Tyler, Texas 75702.

2. Uniloc Luxembourg S.A. ("Uniloc Luxembourg") is a Luxembourg public limited liability company, with its principal place of business at 75, Boulevard Grande Duchesse Charlotte, L-1331, Luxembourg.

3. Uniloc researches, develops, manufactures and licenses information security technology solutions, platforms and frameworks, including solutions for securing software applications and digital content. Uniloc's patented technologies enable software and content

publishers to securely distribute and sell their high-value technology assets with minimum burden to their legitimate end users. Uniloc's technology is used in several markets, including software and game security, identity management, intellectual property rights management, and critical infrastructure security.

4. Electronic Arts, Inc. ("EA" or "Defendant") is a Delaware corporation with its principal place of business in Redwood City, California. EA may be served with process through its registered agent, the Corporation Trust Company, Corporation Trust Center, 1209 Orange St, Wilmington, DE, 19801. Upon information and belief, EA does business in the State of Texas and in the Eastern District of Texas.

JURISDICTION AND VENUE

5. Uniloc brings this action for patent infringement under the patent laws of the United States, namely 35 U.S.C. §§ 271, 281, and 284-285, among others. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331, 1338(a), and 1367.

6. Venue is proper in this judicial district pursuant to 28 U.S.C. §§ 1391(c) and 1400(b). On information and belief, EA is deemed to reside in this judicial district, has committed acts of infringement in this judicial district, has purposely transacted business involving its accused products in this judicial district and/or, has regular and established places of business in this judicial district.

7. EA is subject to this Court's specific and general personal jurisdiction pursuant to due process and/or the Texas Long Arm Statute, due at least to its substantial business in this State and judicial district, including: (A) at least part of its infringing activities alleged herein; and (B) regularly doing or soliciting business, engaging in other persistent conduct, and/or deriving substantial revenue from goods sold and services provided to Texas residents.

COUNT I
(INFRINGEMENT OF U.S. PATENT NO. 5,490,216)

8. Uniloc incorporates paragraphs 1 through 7 herein by reference.

9. Uniloc Luxembourg is the owner, by assignment, of the ‘216 patent, entitled “SYSTEM FOR SOFTWARE REGISTRATION.” A true and correct copy of the ‘216 patent is attached as Exhibit A.

10. Uniloc USA is the exclusive licensee of the ‘216 patent with ownership of all substantial rights in the ‘216 patent, including the right to grant sublicenses, exclude others and to enforce, sue and recover damages for past and future infringements.

11. The ‘216 patent is valid, enforceable and was duly issued in full compliance with Title 35 of the United States Code.

12. EA is directly infringing one or more claims of the ‘216 patent in this judicial district and elsewhere in Texas, including but not necessarily limited to claims 1-5, 7-19, and 19, without the consent or authorization of Uniloc, by or through making, using, offering for sale, selling and/or importing a system, device and/or method for reducing software piracy, reducing casual copying and/or reducing the unauthorized use of software, including without limitation EA’s product activation systems and processes that permit customers to activate and/or register software (the “accused instrumentality”), including but not limited to EA’s SecuROM based product activation systems.

13. Upon information and belief, EA also may be infringing the ‘216 patent through other product activation systems and processes that permit customers to activate and/or register software not presently known to Uniloc, such as but not limited to those product activation systems utilized by games such as (1) Alice: Madness Returns, (2) Dragon Age II, and (3)

Darkspore: Limited Edition. Uniloc reserves the right to discover and pursue relief against all infringing instrumentalities.

14. Uniloc has been damaged as a result of EA's infringing conduct described in this Count. EA is, thus, liable to Uniloc in an amount that adequately compensates it for EA's infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

15. Any allegation of infringement against any defendant herein was not made on the basis of its use, sale, offer for sale, making or importing of any product, software, system, method or service provided by Flexera Software LLC or Rovi Solutions Corporation or any of their present or former affiliates or predecessors (including Flexera Software, Inc. Acresso Software Inc., Installshield Software Corporation, Flexco Holding Company, Inc., Installshield Co Inc., Globetrotter Software, Inc., C-Dilla Limited and Macrovision Corporation) (each a "Licensee Product"), including any product, software, system, method or service incorporating or using the activation, licensing, or registration functionality provided by such Licensee Product.

JURY DEMAND

Uniloc hereby requests a trial by jury pursuant to Rule 38 of the Federal Rules of Civil Procedure.

PRAYER FOR RELIEF

Uniloc requests that the Court find in its favor and against EA, and that the Court grant Uniloc the following relief:

a. Judgment that one or more claims of the '216 patent has been infringed, either literally and/or under the doctrine of equivalents, by EA;

b. Judgment that EA account for and pay to Uniloc all damages to and costs incurred by Uniloc because of EA's infringing activities and other conduct complained of herein;

c. Judgment that EA account for and pay to Uniloc a reasonable, on-going, post judgment royalty because of EA's infringing activities and other conduct complained of herein;

d. That Uniloc be granted pre-judgment and post-judgment interest on the damages caused by EA's infringing activities and other conduct complained of herein; and

e. That Uniloc be granted such other and further relief as the Court may deem just and proper under the circumstances.

Dated: March 21, 2013

Respectfully submitted,

/s/ Edward Casto, Jr. w/permission Wes Hill
Edward E. Casto, Jr.
Lead Attorney
Texas State Bar No. 24044178
Barry J. Bumgardner
Texas State Bar No. 00793424
Steven W. Hartsell
Texas State Bar No. 24040199
Jaime K. Olin
Texas State Bar No. 24070363
R. Casey O'Neill
Texas State Bar No. 24079077
NELSON BUMGARDNER CASTO, P.C.
3131 West 7th Street, Suite 300
Fort Worth, Texas 76107
Phone: (817) 377-9111
Fax: (817) 377-3485
ecasto@nbclaw.net
barry@nbclaw.net
shartsell@nbclaw.net
jolin@nbclaw.net
coneill@nbclaw.net

James L. Etheridge
Texas State Bar No. 24059147
ETHERIDGE LAW GROUP, PLLC
2600 E. Southlake Blvd., Suite 120 / 324
Southlake, Texas 76092
Telephone: (817) 470-7249
Facsimile: (817) 887-5950

Jim@EtheridgeLaw.com

T. John Ward, Jr.
Texas State Bar No. 00794818
J. Wesley Hill
Texas State Bar No. 24032294
WARD & SMITH LAW FIRM
P.O. Box 1231
1127 Judson Road, Ste. 220
Longview, Texas 75606-1231
(903) 757-6400
(903) 757-2323 (fax)
jw@wsfirm.com
wh@wsfirm.com

ATTORNEYS FOR PLAINTIFFS UNILOC USA, INC. AND UNILOC LUXEMBOURG S.A.

Exhibit "A"

US005490216A

United States Patent [19]

Richardson, III

[11] **Patent Number: 5,490,216**

[45] **Date of Patent: Feb. 6, 1996**

[54] **SYSTEM FOR SOFTWARE REGISTRATION**

[75] Inventor: **Frederic B. Richardson, III,** Brookvale, Australia

[73] Assignee: **Uniloc Private Limited**, Singapore

[21] Appl. No.: **124,718**

[22] Filed: **Sep. 21, 1993**

[30] **Foreign Application Priority Data**

Sep. 21, 1992	[AU]	Australia	PL4842
Oct. 26, 1992	[AU]	Australia	PL5524

[51] **Int. Cl.**^{6} **H04L 9/00**

[52] **U.S. Cl.** **380/4**; 380/23

[58] **Field of Search** 380/3, 4, 23, 24, 380/25

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,654,799	3/1987	Ogaki et al. .	
4,688,169	8/1987	Joshi .	
4,796,220	1/1989	Wolfe .	
4,982,430	1/1991	Frezza et al. .	
4,999,806	3/1991	Chernow et al.	380/4
5,191,611	3/1993	Lang	380/4
5,199,066	3/1993	Logan	380/4
5,222,133	6/1993	Chou et al.	380/4
5,239,166	8/1993	Graves .	
5,239,648	8/1993	Nukui	380/4
5,287,408	2/1994	Samson	380/4
5,291,598	3/1994	Grundy .	
5,313,637	5/1994	Rose	380/4
5,319,705	7/1994	Halter et al.	380/4
5,337,357	8/1994	Chou et al.	380/4
5,343,526	8/1994	Lassers	380/4
5,349,643	9/1994	Cox et al.	380/4
5,371,792	12/1994	Asai et al.	380/4
5,379,433	1/1995	Yamagishi	380/4
5,386,468	1/1995	Akiyama et al.	380/4
5,388,211	2/1995	Hornbuckle	380/4
5,390,297	2/1995	Barber et al.	380/4

FOREIGN PATENT DOCUMENTS

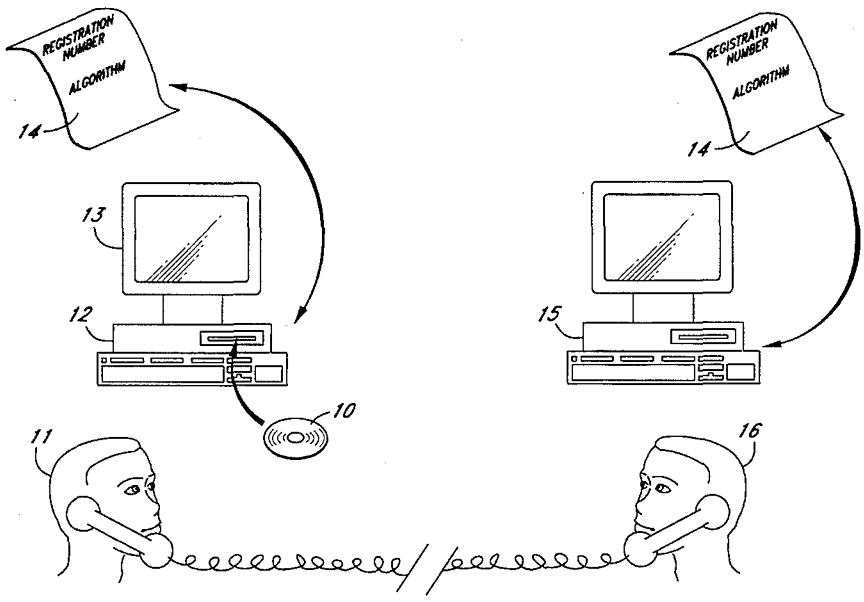
WO9209160	5/1992	WIPO .

Primary Examiner—David C. Cain
Attorney, Agent, or Firm—Knobbe, Martens, Olson & Bear

[57] **ABSTRACT**

A registration system allows digital data or software to run in a use mode on a platform if and only if an appropriate licensing procedure has been followed. Preferably, the system detects when part of the platform on which the digital data has been loaded has changed in part or in entirety, as compared with the platform parameters, when the software or digital data to be protected was last booted or run. The system relies on a portion of digital data or code which is integral to the digital data to be protected by the system. This integral portion is termed the code portion and may include an algorithm that generates a registration number unique to an intending licensee of the digital data based on information supplied by the licensee which characterizes the licensee. The algorithm in the code portion is duplicated at a remote location on a platform under the control of the licensor or its agents, and communication between the intending licensee and the licensor or its agent is required so that a matching registration number can be generated at the remote location for subsequent communication to the intending licensee as a permit to licensed operation of the digital data in a use mode. The code portion can be identical for all copies of the digital data. The algorithm provides a registration number which can be "unique" if the details provided by the intending licenses upon which the algorithm relies when executed upon the platform are themselves "unique".

20 Claims, 12 Drawing Sheets



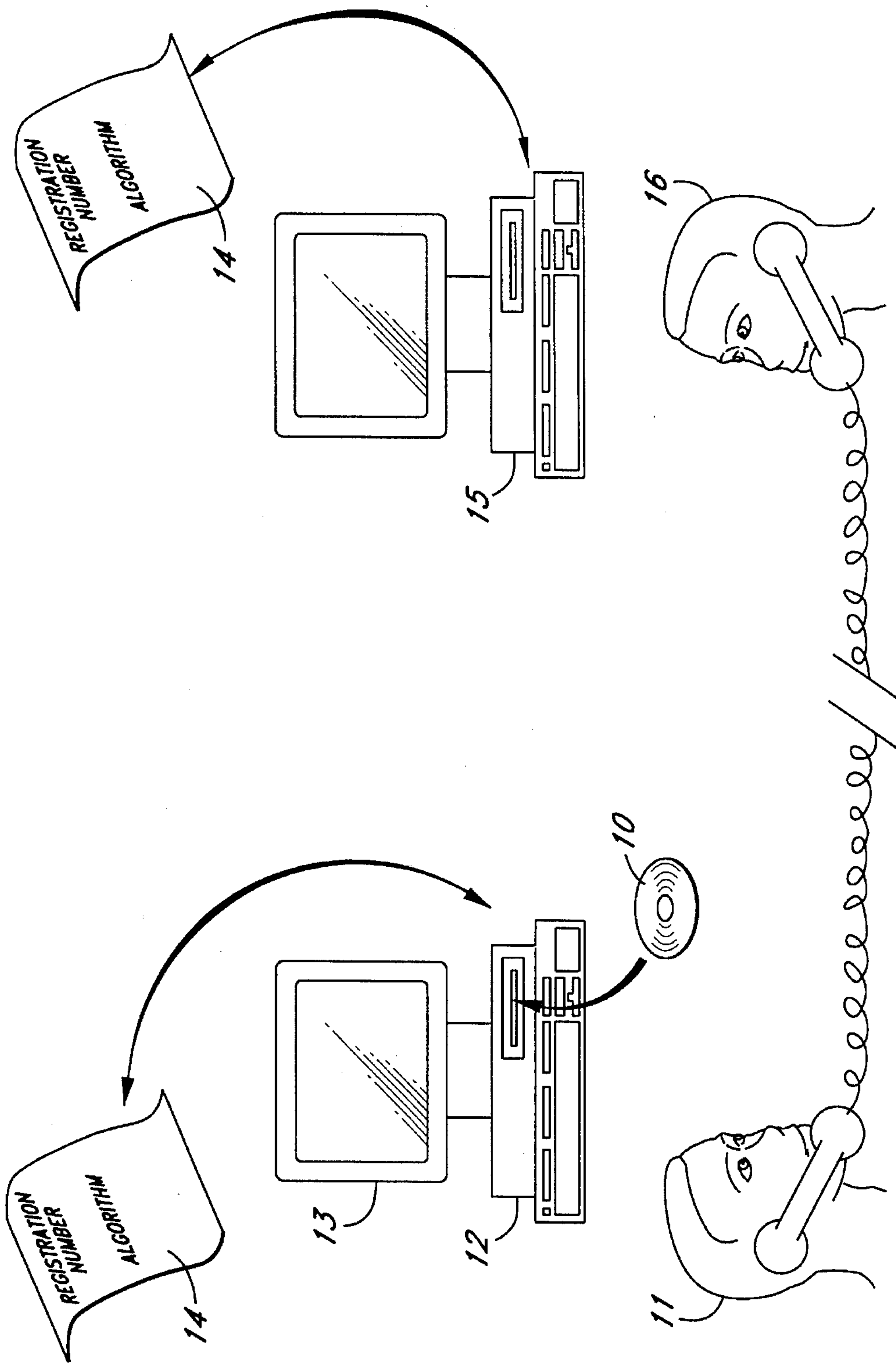

REGISTRATION NUMBER
ALGORITHM
14
15
16
REGISTRATION NUMBER
ALGORITHM
14
13
12
10
11

FIG. 1

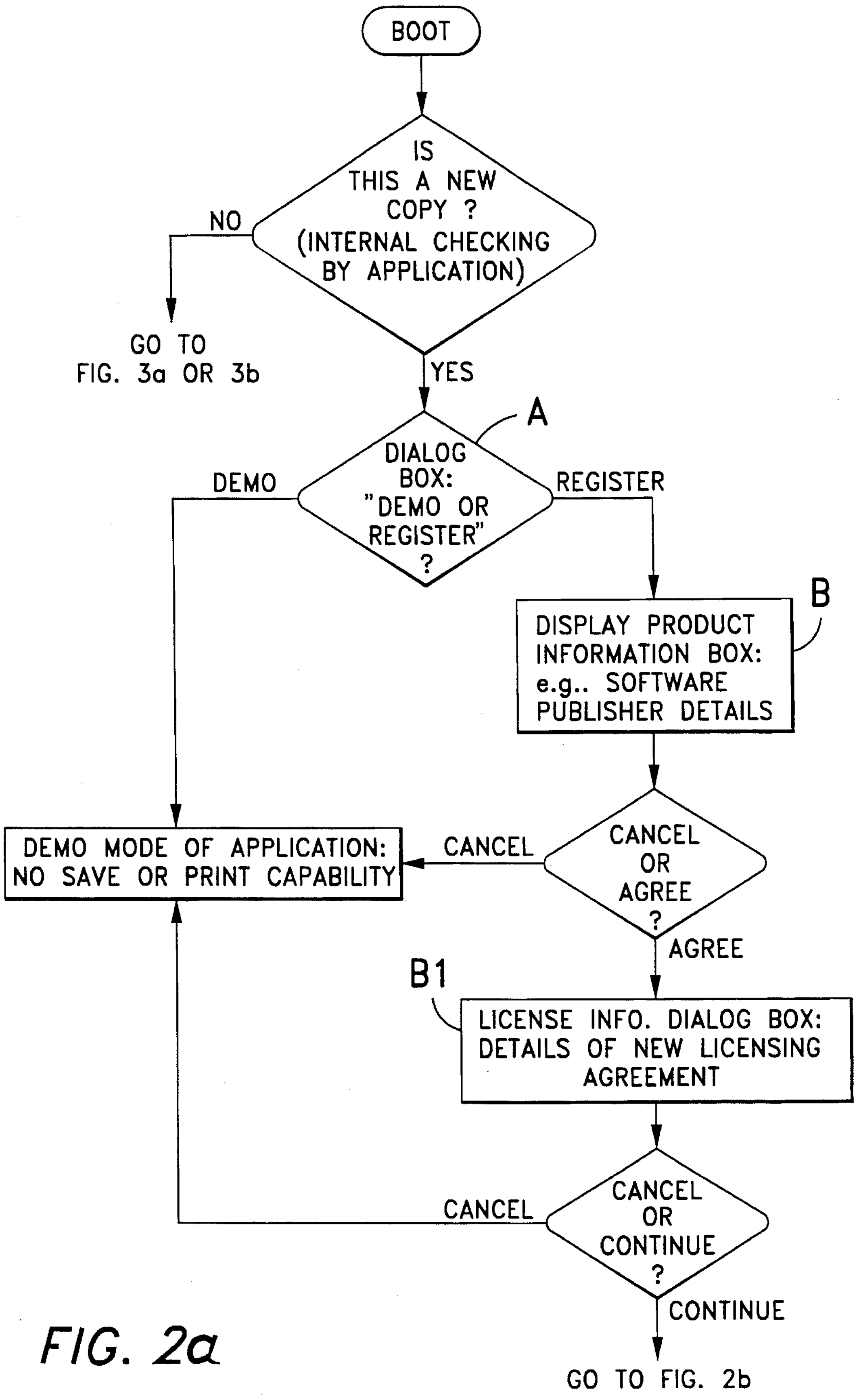

BOOT
IS THIS A NEW COPY ? (INTERNAL CHECKING BY APPLICATION)
NO
GO TO FIG. 3a OR 3b
YES
A
DIALOG BOX: "DEMO OR REGISTER" ?
DEMO
REGISTER
B
DISPLAY PRODUCT INFORMATION BOX: e.g.. SOFTWARE PUBLISHER DETAILS
DEMO MODE OF APPLICATION: NO SAVE OR PRINT CAPABILITY
CANCEL
CANCEL OR AGREE ?
AGREE
B1
LICENSE INFO. DIALOG BOX: DETAILS OF NEW LICENSING AGREEMENT
CANCEL
CANCEL OR CONTINUE ?
CONTINUE
GO TO FIG. 2b

FIG. 2a

FIG. 2b

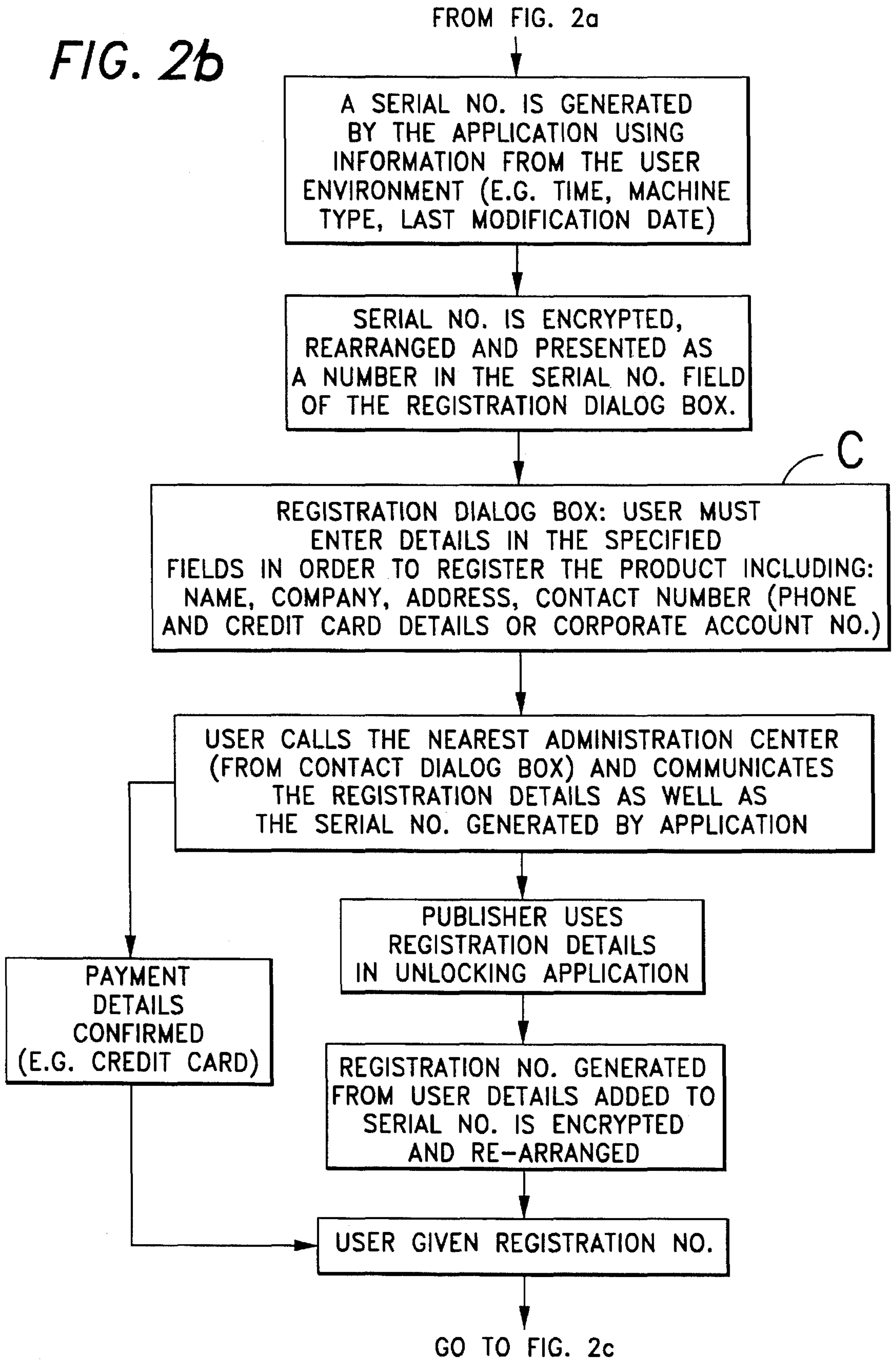

FROM FIG. 2a
A SERIAL NO. IS GENERATED BY THE APPLICATION USING INFORMATION FROM THE USER ENVIRONMENT (E.G. TIME, MACHINE TYPE, LAST MODIFICATION DATE)
SERIAL NO. IS ENCRYPTED, REARRANGED AND PRESENTED AS A NUMBER IN THE SERIAL NO. FIELD OF THE REGISTRATION DIALOG BOX.
C
REGISTRATION DIALOG BOX: USER MUST ENTER DETAILS IN THE SPECIFIED FIELDS IN ORDER TO REGISTER THE PRODUCT INCLUDING: NAME, COMPANY, ADDRESS, CONTACT NUMBER (PHONE AND CREDIT CARD DETAILS OR CORPORATE ACCOUNT NO.)
USER CALLS THE NEAREST ADMINISTRATION CENTER (FROM CONTACT DIALOG BOX) AND COMMUNICATES THE REGISTRATION DETAILS AS WELL AS THE SERIAL NO. GENERATED BY APPLICATION
PUBLISHER USES REGISTRATION DETAILS IN UNLOCKING APPLICATION
PAYMENT DETAILS CONFIRMED (E.G. CREDIT CARD)
REGISTRATION NO. GENERATED FROM USER DETAILS ADDED TO SERIAL NO. IS ENCRYPTED AND RE-ARRANGED
USER GIVEN REGISTRATION NO.
GO TO FIG. 2c

FIG. 2c

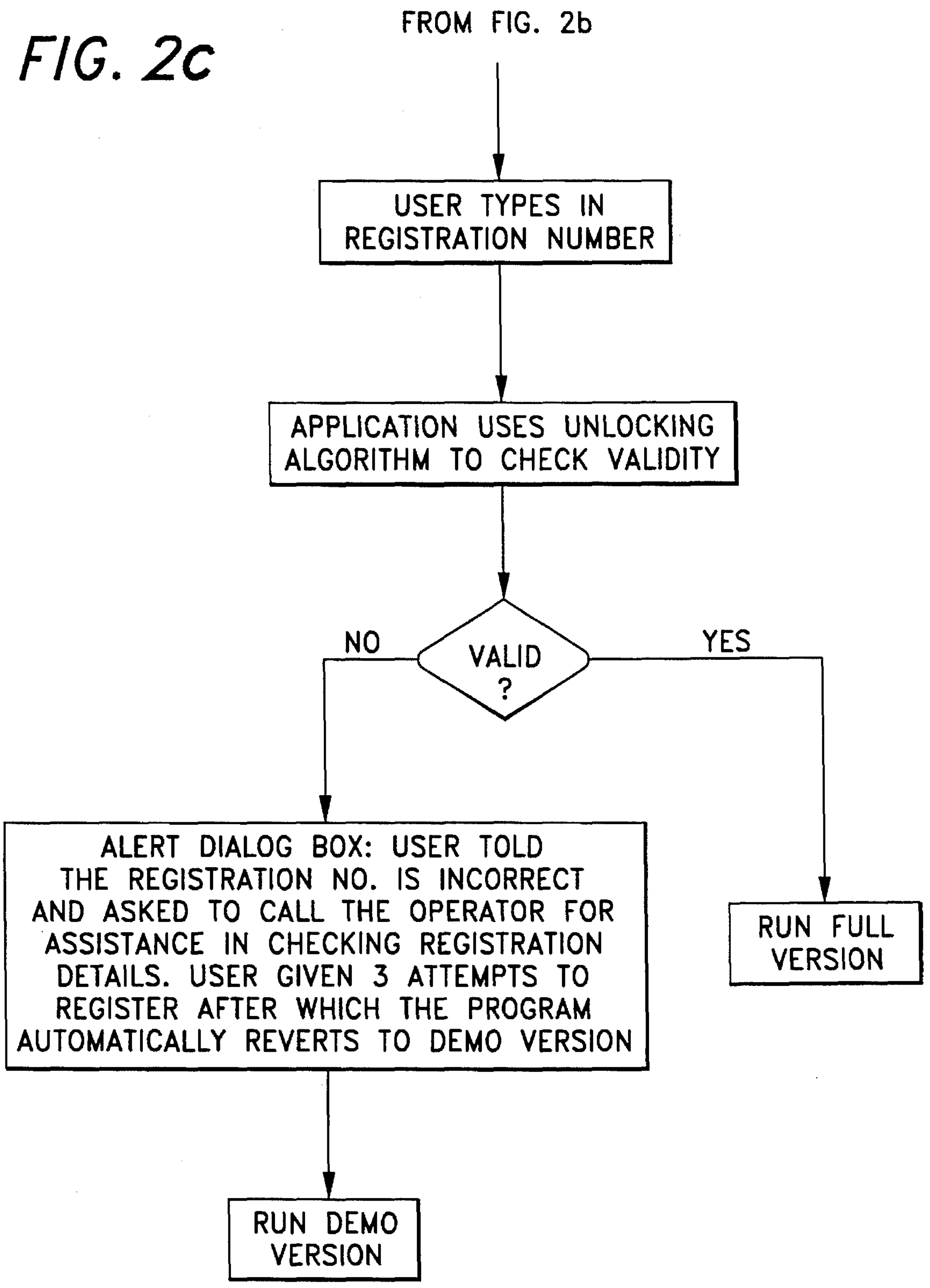
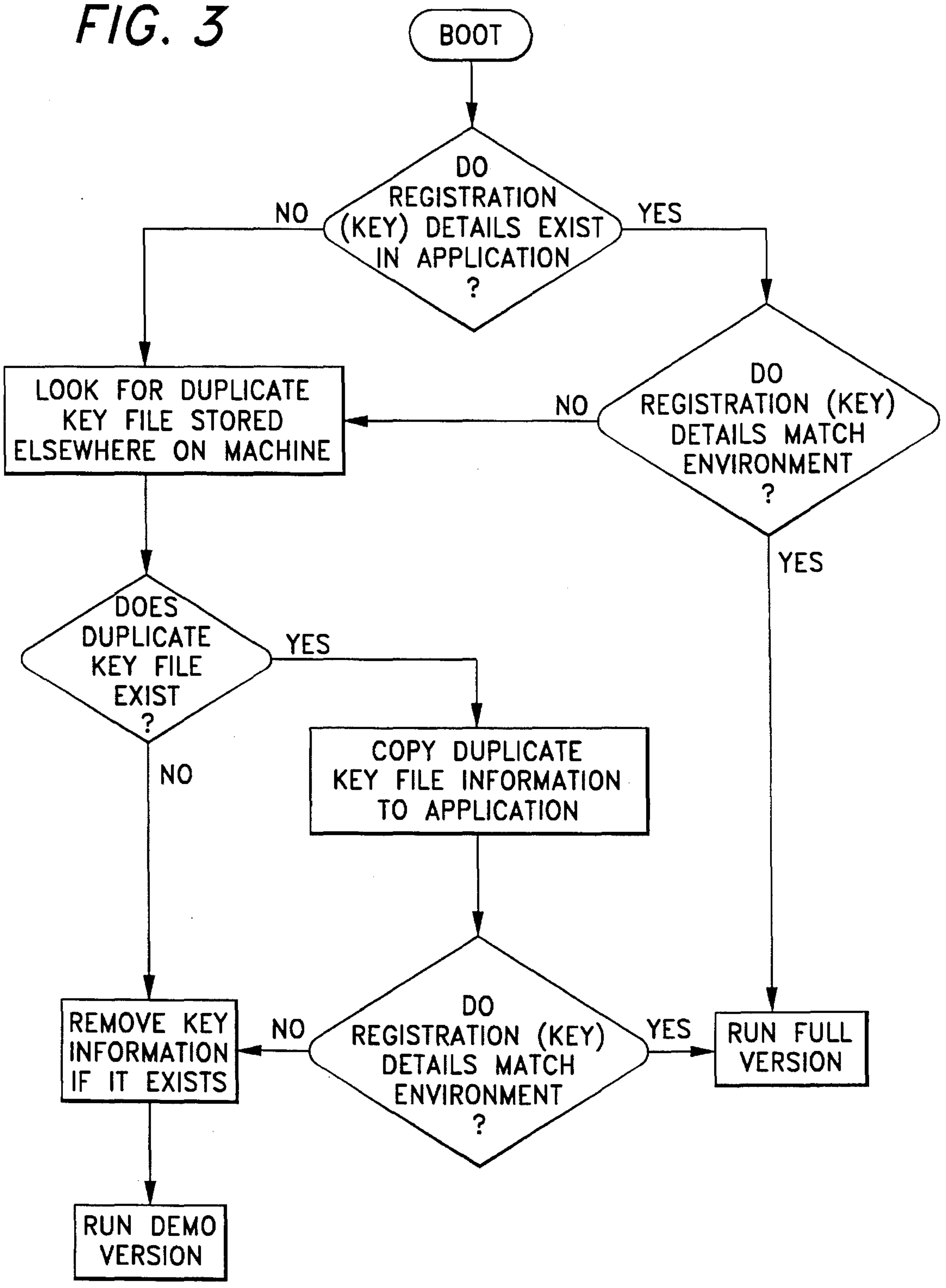

FROM FIG. 2b
USER TYPES IN REGISTRATION NUMBER
APPLICATION USES UNLOCKING ALGORITHM TO CHECK VALIDITY
VALID ?
NO
YES
ALERT DIALOG BOX: USER TOLD THE REGISTRATION NO. IS INCORRECT AND ASKED TO CALL THE OPERATOR FOR ASSISTANCE IN CHECKING REGISTRATION DETAILS. USER GIVEN 3 ATTEMPTS TO REGISTER AFTER WHICH THE PROGRAM AUTOMATICALLY REVERTS TO DEMO VERSION
RUN FULL VERSION
RUN DEMO VERSION

FIG. 3


BOOT
DO REGISTRATION (KEY) DETAILS EXIST IN APPLICATION ?
NO
YES
LOOK FOR DUPLICATE KEY FILE STORED ELSEWHERE ON MACHINE
NO
DO REGISTRATION (KEY) DETAILS MATCH ENVIRONMENT ?
YES
DOES DUPLICATE KEY FILE EXIST ?
YES
NO
COPY DUPLICATE KEY FILE INFORMATION TO APPLICATION
REMOVE KEY INFORMATION IF IT EXISTS
NO
DO REGISTRATION (KEY) DETAILS MATCH ENVIRONMENT ?
YES
RUN FULL VERSION
RUN DEMO VERSION

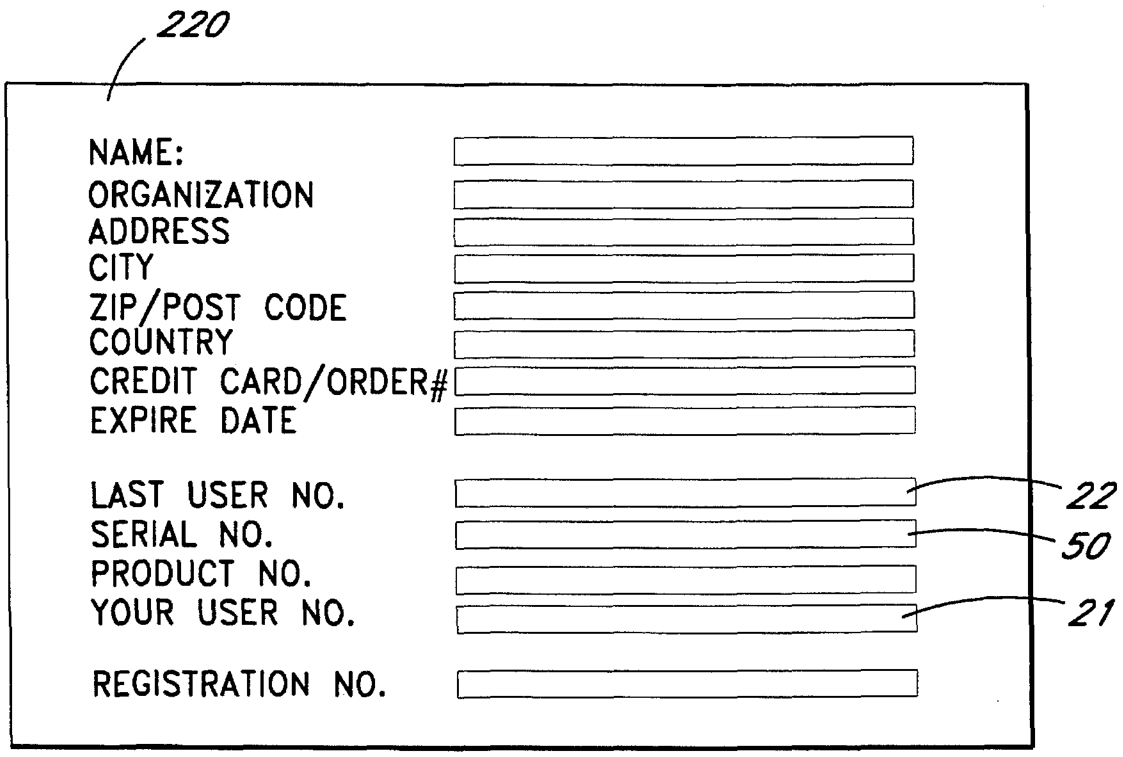

220
NAME:
ORGANIZATION
ADDRESS
CITY
ZIP/POST CODE
COUNTRY
CREDIT CARD/ORDER#
EXPIRE DATE
LAST USER NO.
SERIAL NO.
PRODUCT NO.
YOUR USER NO.
REGISTRATION NO.
22
50
21

FIG. 4

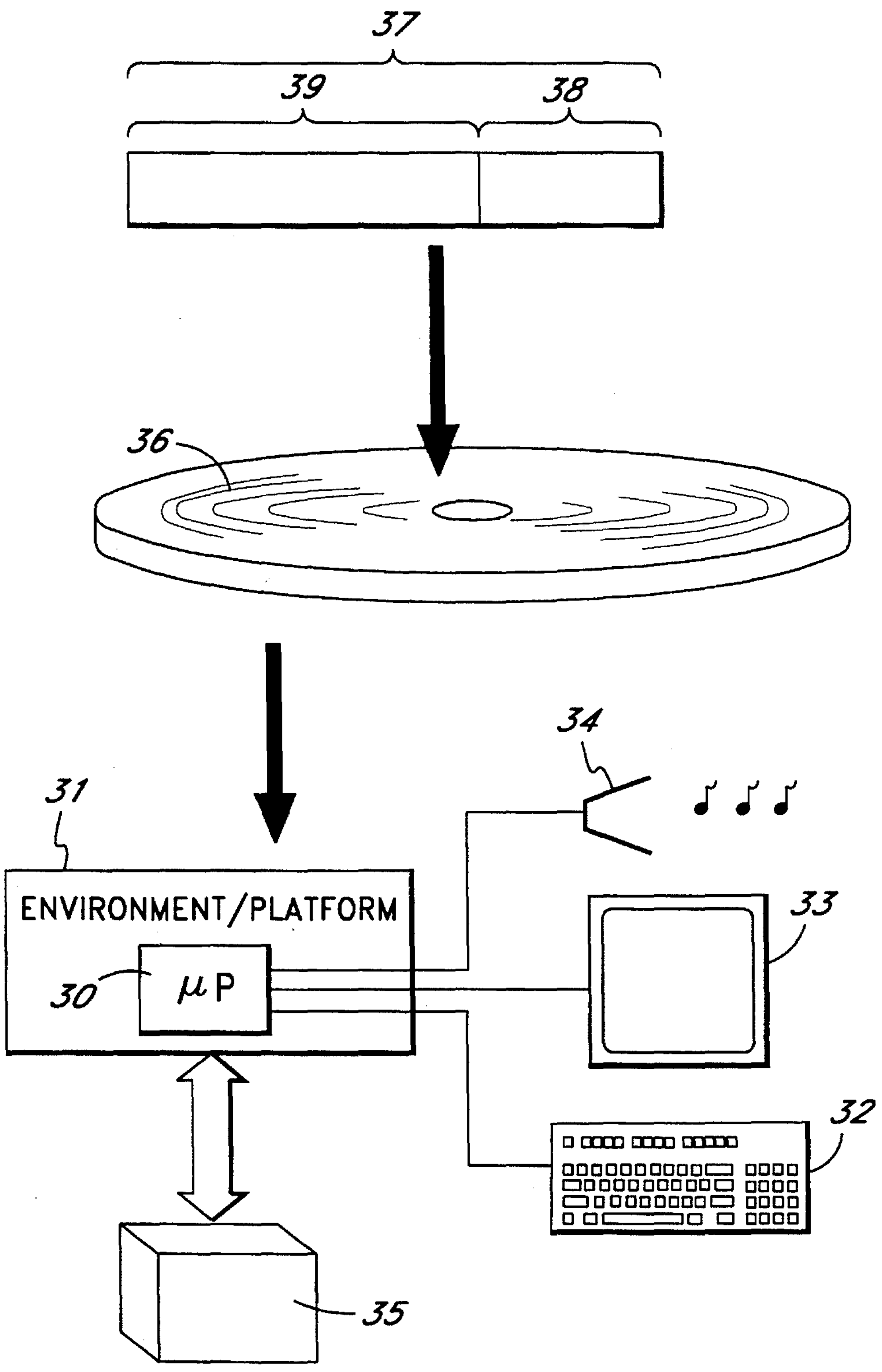


FIG. 5

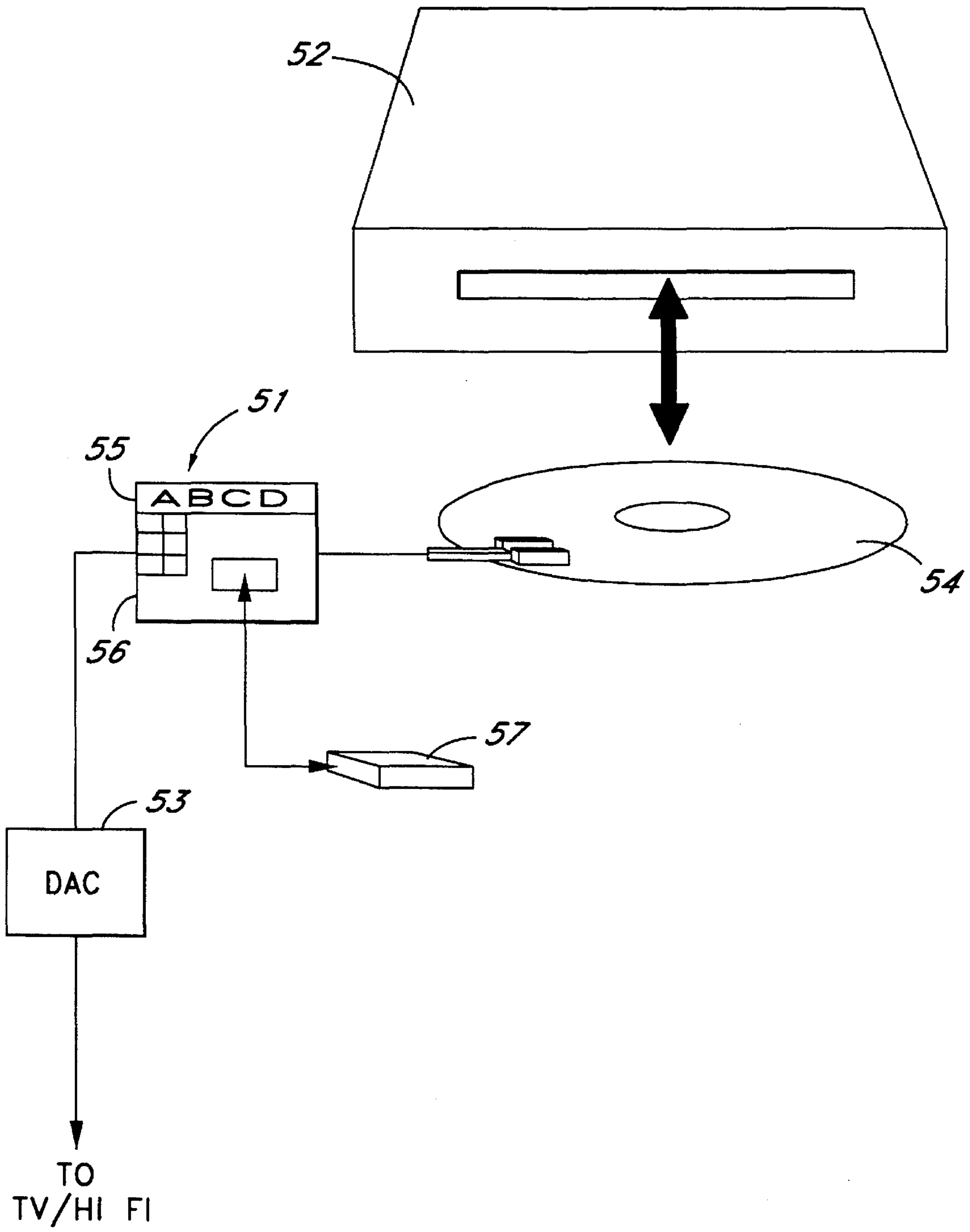


FIG. 6

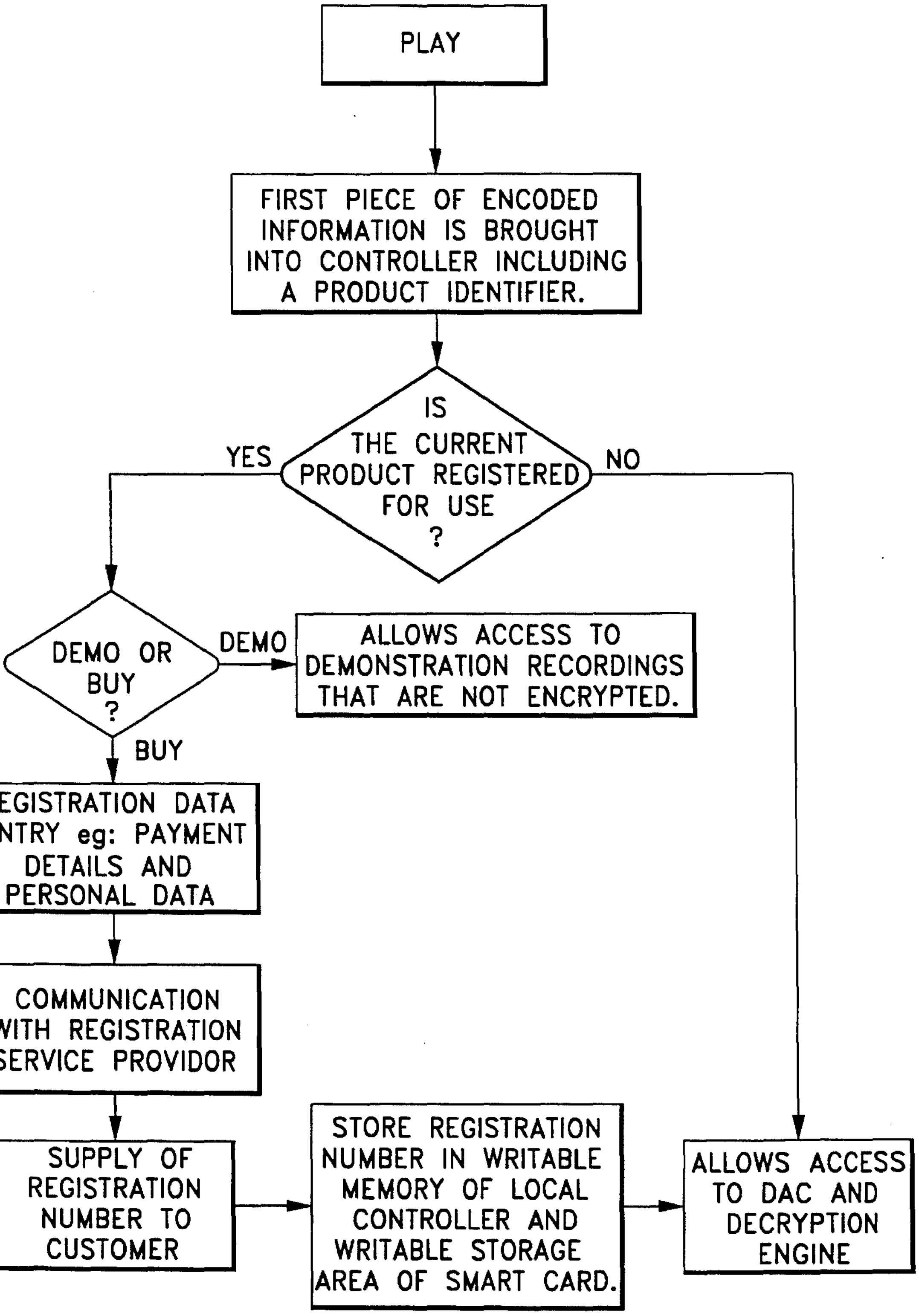

PLAY
FIRST PIECE OF ENCODED INFORMATION IS BROUGHT INTO CONTROLLER INCLUDING A PRODUCT IDENTIFIER.
IS THE CURRENT PRODUCT REGISTERED FOR USE ?
YES
NO
DEMO OR BUY ?
DEMO
ALLOWS ACCESS TO DEMONSTRATION RECORDINGS THAT ARE NOT ENCRYPTED.
BUY
REGISTRATION DATA ENTRY eg: PAYMENT DETAILS AND PERSONAL DATA
COMMUNICATION WITH REGISTRATION SERVICE PROVIDOR
SUPPLY OF REGISTRATION NUMBER TO CUSTOMER
STORE REGISTRATION NUMBER IN WRITABLE MEMORY OF LOCAL CONTROLLER AND WRITABLE STORAGE AREA OF SMART CARD.
ALLOWS ACCESS TO DAC AND DECRYPTION ENGINE

FIG. 7

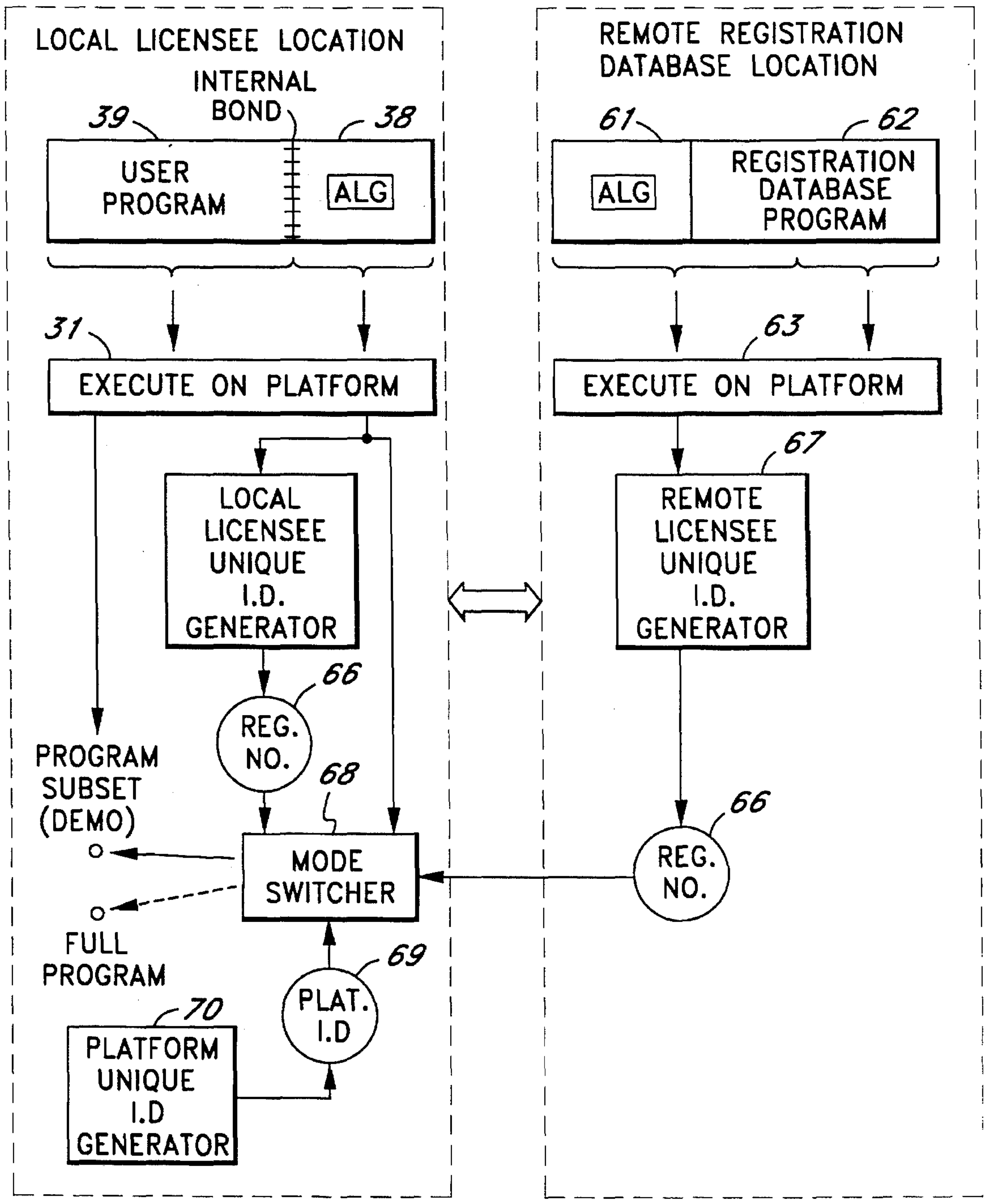

LOCAL LICENSEE LOCATION
INTERNAL BOND
39
38
USER PROGRAM
ALG
31
EXECUTE ON PLATFORM
LOCAL LICENSEE UNIQUE I.D. GENERATOR
66
REG. NO.
68
PROGRAM SUBSET (DEMO)
MODE SWITCHER
FULL PROGRAM
69
PLAT. I.D
70
PLATFORM UNIQUE I.D GENERATOR
REMOTE REGISTRATION DATABASE LOCATION
61
62
ALG
REGISTRATION DATABASE PROGRAM
63
EXECUTE ON PLATFORM
67
REMOTE LICENSEE UNIQUE I.D. GENERATOR
66
REG. NO.

FIG. 8

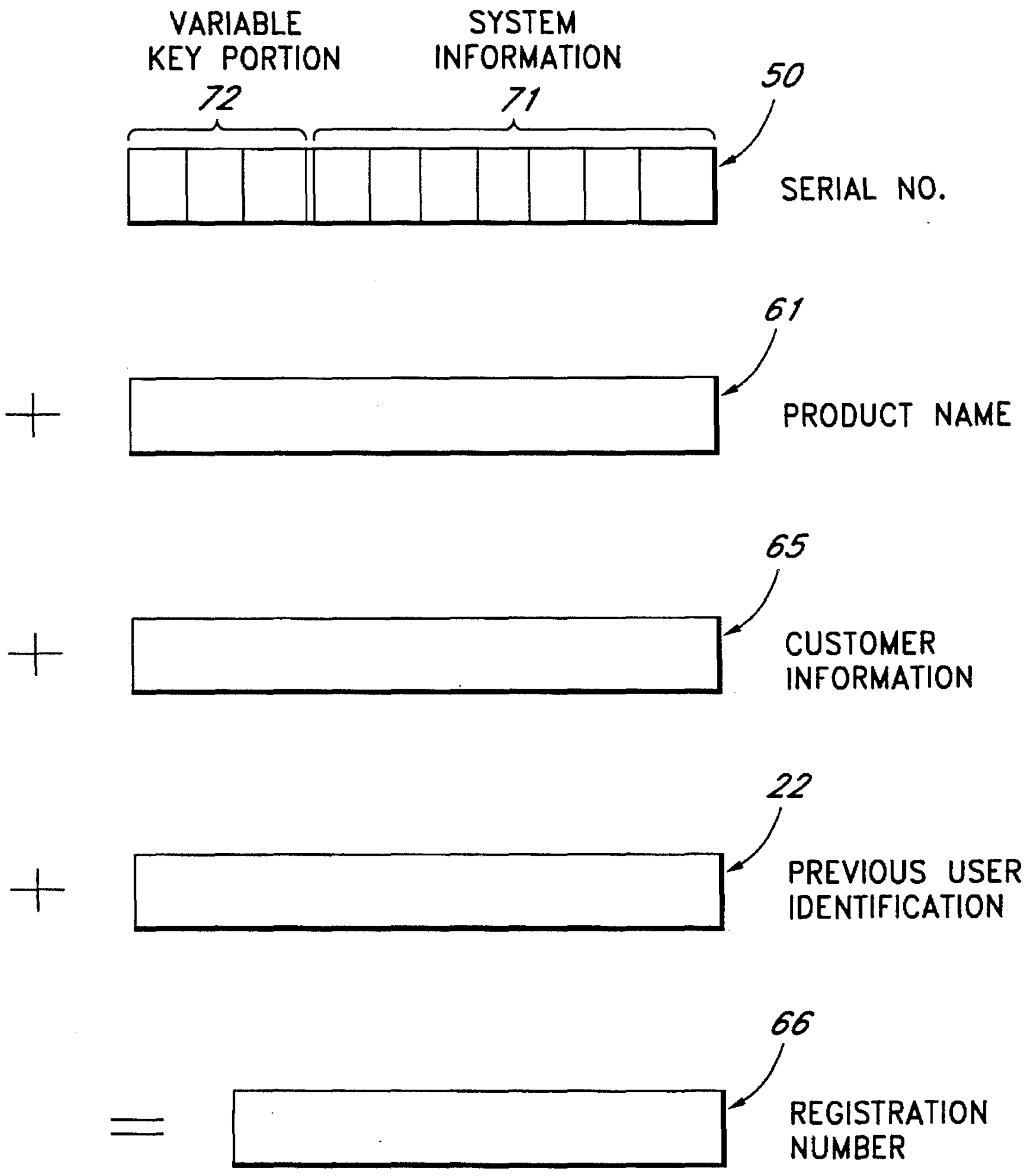

VARIABLE KEY PORTION
72
SYSTEM INFORMATION
71
50
SERIAL NO.
61
PRODUCT NAME
65
CUSTOMER INFORMATION
22
PREVIOUS USER IDENTIFICATION
66
REGISTRATION NUMBER

FIG. 9

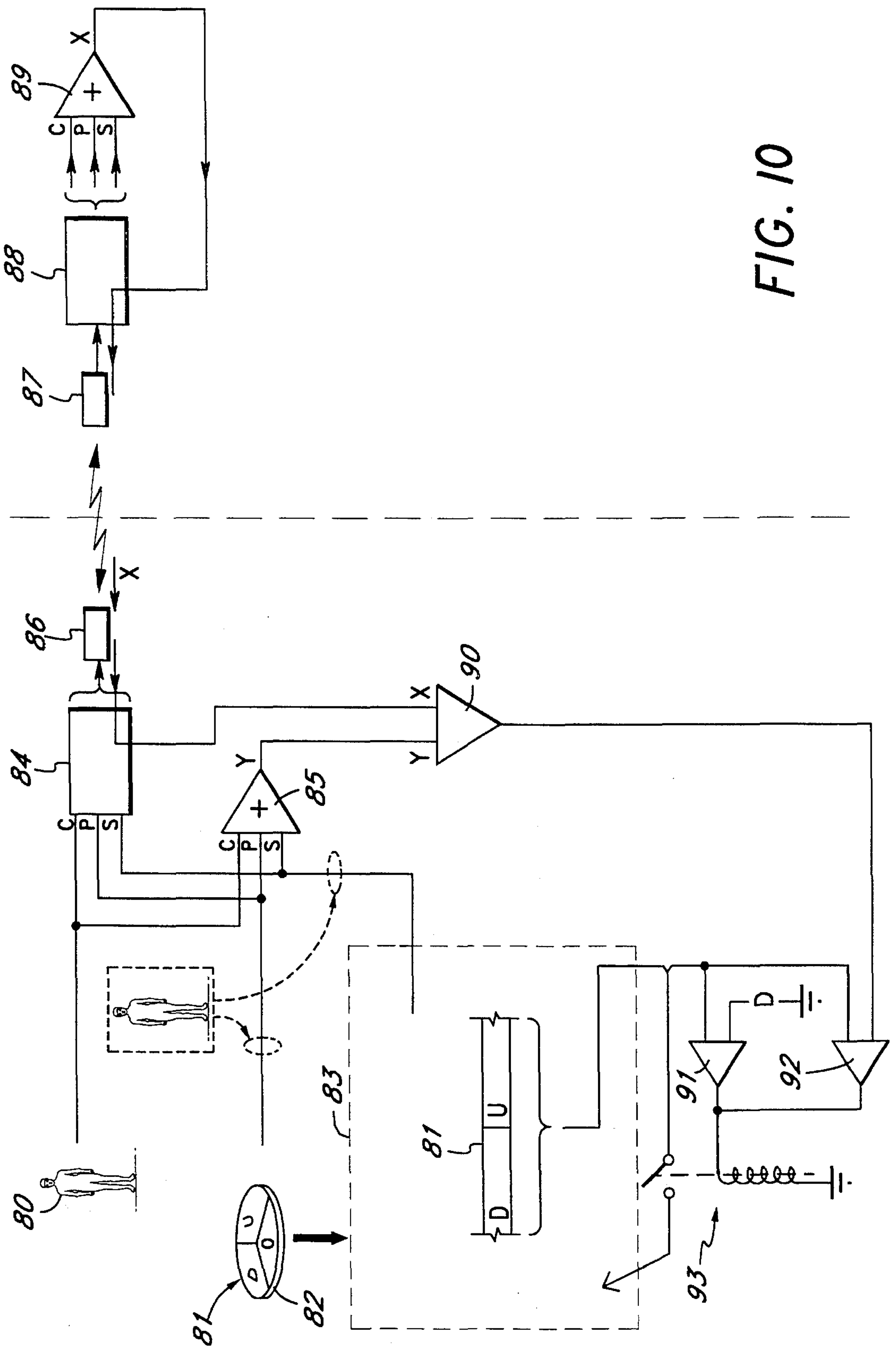

89
X
+
C
P
S
88
87
86
X
84
C
P
S
85
+
Y
C
P
S
90
X
Y
83
81
U
D
80
81
U
O
D
82
D
91
92
93

FIG. 10

SYSTEM FOR SOFTWARE REGISTRATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to improvements in systems for software registration and, more particularly, to improvements in arrangements where software is transferable by media such as magnetic disks, CD ROMS and the like.

2. Description of the Related Art

Much commercially available software is provided at time of purchase (or license) on a magnetic media, typically a floppy disk. Frequently the only security feature attached to the software is a simple registration number stored on the media. This registration number identifies that particular copy of the software and it is often required at the time of installation of the software onto any given computer that the installer must provide the registration number independently to the installation routines.

However, such simple security arrangements for the distribution of software on media suffer from at lest two disadvantages: (1) each copy of the software made on any given media at the time of manufacture must include an individual, unique number, programmed into the media, and (2) this arrangement does not prevent copying of the software, once installed on any given computer, to another computer by means of file transfer (as opposed to reinstallation).

WO 92/09,160 to Tan Systems Corporation discloses a registration system which is relatively sophisticated which relies for its security on a requirement that an intending software licensee must obtain from a remote location by file transfer significant and essential portions of the program which the licensee desires to execute. The arrangement disclosed in WO 92/09,160 suffers from a number of deficiencies including:

a. the shell program which the intending licensee initially executes requires a unique identity embodied within the shell prior to distribution of the shell program;

b. the shell program is not, itself, a functional program—that is, it does not include all of the code which the intending licensee wishes to execute. That program must be obtained remotely with all the delays, inconveniences and possibilities of corruption during transit that that entails;

c. the prior art system appears to require and indeed, rely on, encryption to ensure that the program material which is communicated from a remote location is not intercepted for utilization in an unauthorized manner; and

d. it is unclear whether the system can accommodate and react appropriately to the situation where the program, once registered, is transferred in its entirety from one platform to another so as to avoid the requirement for payment of a further registration fee.

U.S. Pat. No. 4,796,220, assigned to Pride Software Development Corporation, discloses a system for unique recognition of a platform on which licensed software is to be executed. However, U.S. Pat. No. 4,796,220 does not contemplate or disclose utilization of information which is unique to the user or intended licensee as part of the registration process which is to be distinguished from identification of the platform upon which the software is proposed to be run.

U.S. Pat. No. 4,688,169 to Joshi broadly discloses the same principles as U.S. Pat. No. 4,796,220 in that it discloses a computer software security system which relies for its security on a "machine identification code unique to the machine" upon which the software to be protected is to be run. Again, the disclosure is limited to identification of the platform and there is no suggestion or contemplation of linking platform identification with unique user identification.

Also this arrangement does not allow the flexibility of transfer of copies of the program from platform to platform which can be run in a demonstration mode.

It is an object of the present invention to address or reduce the above-mentioned disadvantages.

Definitions

Throughout this specification the term "software" is to be interpreted broadly so as to include all forms of digital data which are executable on a platform (as to be later defined). The digital data comprising the software can, for example, be code comprising a word processing program adapted to run on a PC or the like. The software can also, for example, be digital data stored on a CD ROM adapted for playback as music on a CD ROM audio drive. The digital data can be displayable information or information which is otherwise usable by a licensed user.

Throughout this specification the term "platform" denotes an environment to be associated with a computing device such as a microprocessor or other data processing device which permits execution of the digital data (to which reference has previously been made in relation to the term "software") whereby the computer can perform functions on input and output devices associated therewith.

In some circumstances, the "software" or digital data may itself be the operating system environment. Typically, but by no means exclusively, examples of operating system environments include the MicroSoft DOS operating system, the IBM OS/2 operating system or the Macintosh System 7 environment. In the degenerate case of microcontrollers operating from ROM, the operating system environment may be the microcode of the microcontroller which enables the microcontroller to execute machine code.

In this specification, "use mode" refers to use of the digital data or software by its execution on a platform so as to fulfill the seller's/licensor's obligations in relation to the sale or license of the right to execute the digital data or software in the use mode. The use mode is to be distinguished from what might generally be termed unlicensed modes of operation (which is not to say unauthorized modes of operation) as typified by the demonstration modes later described in this specification.

SUMMARY OF THE INVENTION

In broad terms, the system according to the invention is designed and adapted to allow digital data or software to run in a use mode on a platform if and only if an appropriate licensing procedure has been followed. In particular forms, the system includes means for detecting when parts of the platform on which the digital data has been loaded has changed in part or in entirety as compared with the platform parameters when the software or digital data to be protected was for example last booted or run or validly registered.

The system relies on digital data or code which forms part of the digital data to be protected by the system. This portion of the digital data which preferably is integral to the digital data to be protected has been termed the "code portion" elsewhere in this specification. The code portion includes an algorithm adapted to generate a registration number which is unique to an intending licensee of the digital data based on information supplied by the licensee which characterizes the licensee.

The algorithm in the code portion is duplicated at a remote location on a platform under the control of the licensor or its agents and communication between the intending licensee and the licensor or its agent is required so that a matching registration number can be generated at the remote location for subsequent communication to the intending licensee as a permit to licensed operation of the digital data in a use mode.

Preferably, the code portion is integral with the digital data and can be identical for all copies of the digital data. It is the algorithm embedded within the code portion (and which is duplicated at the remote location) which provides a registration number which can be "unique" if the information provided by the intending licensee upon which the algorithm relies when executed upon the platform is itself "unique."

In any event, in particular preferred forms, a serial number (see further on) is included in the registration number generation algorithm which introduces an additional level of uniqueness into the registration number calculation process.

Accordingly, in one broad form of the invention there is provided a system for licensing use of digital data in a use mode, the digital data executable on a platform, the system including local licensee unique ID generating means and remote licensee unique ID generating means, the system further including mode switching means operable on the platform which permits use of the digital data in the use mode on the platform only if a licensee unique ID generated by the local licensee unique ID generating means has matched a licensee unique ID generated by the remote licensee unique ID generating means.

Preferably, the system further includes platform unique ID generating means, wherein the mode switching means will permit the digital data to run in the use mode in subsequent execution of the digital data on the platform only if the platform unique ID has not changed.

Preferably, the mode switching means permits operation of the digital data in the use mode in subsequent execution of the digital data only if the licensee unique ID generated by the local licensee unique ID generating means has not changed.

Preferably, the mode switching means includes part of the digital data.

Preferably, the remote licensee unique ID generating means comprises software which includes the algorithm utilized by the local licensee unique ID generating means to produce the licensee unique ID.

Preferably, the information utilized by the local licensee unique ID generating means to produce the licensee unique ID comprises prospective licensee credit card number, date of birth and full name and address.

Preferably, the platform unique ID generating means forms part of the digital data.

Preferably, the platform unique ID generating means utilizes hard disk information and/or other computer hardware or firmware information to determine the platform unique ID.

Preferably, the platform comprises a computer operating system environment.

Preferably, the digital data comprises a software program adapted to run under the operating system environment.

In a further broad form of the invention, there is provided a security routine or registration means attachable to software to be protected, the registration means generating a security key from information input to the software which uniquely identifies an intended registered user of the software on a computer on which the software is to be installed.

Preferably, the security key is generated by a registration number algorithm.

Preferably, the registration number algorithm combines information entered by a prospective registered user unique to that user with a serial number generated from information provided by the environment in which the software to be protected is to run (e.g., system clock, last modify date, user name).

Preferably, the registration means is replicated at a registration authority and used for the purposes of checking by the registration authority that the information unique to the user is correctly entered at the time that the security key is generated by the registration means.

Preferably, the registration means checks at the time of boot of the software as to whether it is a first boot of the software to be protected or a subsequent boot. If a subsequent boot is detected, then environment and user details are compared to determine whether the program reverts to a demonstration mode and a new user registration procedure is to commence, or a full version run.

Preferably, the environment details comprise one or more of disc volume name, user name or computer, initialization date of hard disk, hardware identifier (e.g., ROM checksum) or other elements which are generally not user-configurable on the platform.

In a further broad form of the invention, there is provided a method of control of distribution of software, the method comprising providing mode-switching means associated with the software adapted to switch the software between a fully enabled mode and a partly enabled or demonstration mode; the method further comprising providing registration key generating means adapted to generate an enabling key which is a function of information unique to an intending user of the software; the mode-switching means switching the software into fully enabled mode only if an enabling key provided to the mode-switching means by the intending user at the time of registration of the software has matched identically with the registration key generated by the registration key generating means.

Preferably, the enabling key is communicated to the intending user at the time of registration of the software by a third party operating a duplicate copy of the registration key generating means.

In yet a further broad form of the invention, there is provided digital data incorporating registration code, the digital data executable on a platform; the registration code comprising a portion of the digital data executable on the platform so as to switch the digital data between a demonstration mode and a use mode.

Preferably, the registration code when executed on the platform provides local licensee unique ID generating means whereby the digital data can be switched from the demonstration mode to the use mode by execution of the registration code only if a licensee unique ID generated by the local licensee unique ID generating means has matched a licensee unique ID generated by remote licensee unique ID generating means.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described with reference to the accompanying drawings wherein:

FIG. **1** is a schematic diagram of the relationship and interaction between an intending registered user and a registration authority of software on media secured according to a first embodiment of the invention;

FIGS. **2***a*, **2***b* and **2***c* are segments of a flow chart of the procedure to be followed during registration of software by a user according to a first embodiment of the invention;

FIG. **3** is a flow chart of alternative boot processes according to a second embodiment of the invention;

FIG. **4** is a personal information dialogue box relating to the procedure of FIGS. **2***a*, **2***b*, **2***c* in accordance with a third embodiment;

FIG. **5** is a schematic diagram of a system according to a fourth embodiment of the invention;

FIG. **6** is an implementation of the fourth embodiment of FIG. **5** in relation to a CD ROM drive;

FIG. **7** is a logic flow chart in relation to the decoder box of FIG. **6**;

FIG. **8** is a block diagram of a generalized system according to a fifth embodiment of the invention;

FIG. **9** is a block diagram indicating one particular example of generation of a registration number for the system of FIG. **8**; and

FIG. **10** is a schematic diagram of a sixth embodiment comprising a particular example of the generalized system of FIG. **8**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

It is to be understood that, in its various embodiments, the present invention is for the protection of digital code/software by control of permission to use the digital code/software. A hardware platform and a remote registration station implemented at least partially by means of electronic hardware are required by the various embodiments.

The code/software to be protected requires at least some adaption to be usable with the invention in its various embodiments. The adaptation can be universal for all copies of the code/software to be protected.

First Embodiment

With reference to FIGS. **1** and **8**, the system according to embodiments of the invention is designed and adapted to allow digital data **39** or software to run in a use mode on a platform **31** if and only if an appropriate licensing procedure has been followed. In particular forms, the system includes means for detecting when parts of the platform **31** on which the digital data **39** has been loaded has changed in part or in entirety as compared with the platform parameters when the software or digital data to be protected was, for example, last booted or run or validly registered.

The system relies on digital data or code **38** which forms part of the digital data to be protected by the system. This portion of the digital data, which preferably is integral to the digital data to be protected, has been termed the code portion **38** elsewhere in this specification. The code portion **38** includes an algorithm adapted to generate a registration number **66** or local licensee unique ID or registration key which characterizes the licensee. In this instance, the local licensee unique ID generator which generates the registration number comprises the execution of code **38** on platform **31**.

The algorithm in the code portion is duplicated at a remote location on a platform **67** under the control of the licensor or its agents, and communication between the intending licensee and the licensor or its agent is required so that a matching registration number or enabling key can be generated at the remote location for subsequent communication to the intending licensee as a permit to licensed operation of the digital data **39** in a use mode.

Execution of the duplicated code portion on platform **67** comprises, in this instance, the remote licensee unique ID generating means.

Mode switching means can comprise execution of the code portion which additionally performs a comparison of the locally and remotely generated registration numbers.

Preferably, the code portion **38** is integral with the digital data and can be identical for all copies of the digital data. It is the algorithm embedded within the code portion (and which is duplicated at the remote location) which provides a registration number which can be "unique" if the information provided by the intending licensee upon which the algorithm relies when executed upon the platform is itself "unique".

In any event, in particular preferred forms, a serial number (see further on) is included in the registration number generation algorithm which introduces an additional level of uniqueness into the registration number calculation process.

With particular reference to FIG. **1**, a program comprising digital data protected according to a first embodiment of the invention is supplied recorded on a magnetic disk **10**.

Included as part of the software on that disk **10** is a registration and re-registration routine which executes whenever the program protected by the arrangement of the first embodiment "boots".

With reference to FIG. **1** and FIGS. **2***a*, **2***b* and **2***c*, the operation of the security routine will be described on the assumption that the program on the disk **10** protected by the registration routine has not been registered on the platform or is otherwise being loaded for the first time.

The prospective new user **11** inserts disk **10** into the user PC **12** so as to be read by PC **12**.

As part of the software installation procedure, the registration routine is activated causing a series of dialogue boxes to appear on the display **13** of the user PC **12**. Having checked to ensure that the software has not previously been registered on the PC **12**, a dialogue box A (in FIG. **2***a*) is displayed which provides the user with a choice of either seeing a demonstration of the software (which typically has features such as save and/or print disabled) or alternatively an invitation to register ownership/licensee of the software (after which all features of the software are made available to the user).

If the register option is selected or if the user cancels the demonstration in favor of registration, then a contact dialogue box B (in FIG. **2***a*) is presented on the display **13** which provides a list (stored on disk **10** as part of the registration routine) which provides for example, names and contact numbers of the software publishing company together with other general product information.

Following the user's indication of agreement during display of license details (box **B1**) to proceed to register, the user can contact the registration center after filling out the registration dialogue box C as detailed below. After selecting "continue", the registration routine begins the first step in the generation of a security key which will be unique to the current copy of the software and to certain features of the environment in which it runs.

As shown in FIG. **2***b*, the first step in the generation of the security key comprises the generation of a serial number generated from the current time on the system and, in this example, the last modify date of the software and other information from the computer environment. The serial number is encrypted and rearranged and then presented as a number in the registration dialogue box on the display **13**.

The registration dialogue box C (in FIG. **2***b*) prompts the user for details unique to that user (including, for example, name, company, address, state, contact number) together with financial details for payment for the purpose of becoming a registered user of the software protected by the registration routine (for example Mastercard or corporate account number details). This information, unique to the user, is passed through a registration number algorithm **14** (represented symbolically in FIG. **1**) which generates a registration number or security key from the information unique to the user together with the serial number previously generated. The registration number or security key is not made available to the user of the PC **12** by the PC **12**.

An identical registration number algorithm **14** resides on the registration authority PC **15**. As an integral part of the registration procedure, the prospective new user **11** communicates the information unique to the user which was entered by the user on the user PC **12**, along with the serial number generated by the user's algorithm, to the registration authority **16**. The registration authority feeds this information into the registration authority PC **15** wherein the registration number algorithm **14** should produce an identical registration number or security key to that produced by the user PC **12** if the details communicated to the registration authority by the prospective new user **11** match with the details that have been entered on the user PC **12**. Optionally, the user can communicate the information to the registration authority electronically, e.g., by fax or modem or tone phone.

As a final stage in registration (refer to FIG. **2***d*), the registration authority **16** provides the registration number generated by the registration authority PC **15** to the user **11**. The user **11** enters the registration number into the user PC **12** where the registration routine checks to see whether the entered registration number matches the calculated registration number. If the two match, then a valid registration has taken place and access is provided by the registration routine to a full operating version of the software protected by the registration routine. If there is no match and a preference file (which stores the user details) does not exist then a dialogue box D (FIG. **2***c*) appears on the display **13** of user PC **12** providing the prospective new user **11** with the opportunity to check his/her details or switch to the demonstration version of the software protected by the registration routine.

Again, the registration authority PC **15** can provide to PC **12** the registration number which it generates by electronic means such as modem communication.

It will be evident that it is not obvious to the prospective new user **11** that the registration number which unlocks the full version of the software protected by the registration routine is, in fact, generated from an algorithm residing on the magnetic disk **10** and that it forms part of the software to which access is desired.

In this manner, the registration procedure outlined above ensures that exactly the same details entered by the prospective new user on his/her user PC **12** are those details recorded by the registration authority **16**. It will also be evident that the procedure does not require each magnetic disk **10** containing a copy of the software to be protected to have a unique registration number recorded on the disk at the time of distribution of the disk. Each copy has exactly the same registration number algorithm located upon it. A unique registration number or "security key" is generated only at the time of registration from the details supplied by the prospective new user **11**.

The registration routine behaves generally as follows where any copy of the protected software boots. In this situation, the registration routine checks at the time of boot to see what registration details are present for that particular copy of the software. If no details are present, then it is assumed that the PC is booting from a newly distributed magnetic disk and registration is to occur for the first time. The registration procedure in that case is that followed in respect of FIGS. **2***a*, **2***b* and **2***c*.

In the event that registration details are present, then the registration routine checks a number of parameters which are expected to be unique to the environment in which the software to be protected operates. In this embodiment, the parameters checked are hard disk volume name, user name, and computer name and user password and hard disc initialization date (not generally user configurable on the Apple Macintosh computer). The registration routine then checks these parameters against the corresponding details that it finds from the operating environment of the computer on which the software is running. If a designated combination of these details matches then it is assumed that a properly authorized and registered copy of the software is running and full access to the software is allowed.

In this manner, it is quite in order for users to provide other users with copies of the software protected by the security routine. The security routine attached to the software to be protected determines from the environment in which it operates whether an additional registration fee is required. If it is determined by the registration routine that this is the case, then the registration routine has the capability to provide a fresh registration number as part of an authorized registration procedure pending which the protected software reverts to demonstration mode.

Second Embodiment

(Auto re-registration)

According to a second embodiment, a more sophisticated procedure suitable for checking at first boot and at subsequent boot is shown in flowchart form in FIG. **3**.

This procedure incorporates redundancy to cope with situations where the key file containing the information from which the current use has been authorized may have been deleted or does not exist on a subsequent boot.

The distinction as against the first embodiment is that a "key file" is created at the time of registration of the software and a duplicate key file is also created at the same time. The duplicate key file is arranged to be stored on the computer at a location separate from the program to be protected. In the case of the Apple Macintosh computer the duplicate key file can be stored in the "system" folder.

Both the key file (stored with the software) and the duplicate key file are encrypted and both contain identical information. The information contained comprises:

1. The user registration details including the serial number,
2. The environment details of the computer, and
3. Details of the application protected by the security routine for which registration is to be or has been obtained.

With reference to FIG. **3**, whenever the protected application boots, a check is made by the registration routine to determine whether registration details exist in the key file of the protected application. If they do, a comparison is made by the registration routine between what is stored in the key file and the environment to determine whether a change has taken place to the environment as compared with what is stored in the key file. If no change is detected, then the protected application is permitted to run normally.

If there are no registration details present in the key file or if the above-referenced comparison between the key file contents and the application does not show a match, then the re-registration routine of FIG. **3** looks for the existence of a duplicate key file within the environment. If a duplicate key file exists, then the information contained within that duplicate key file is copied to the application key file and comparisons as previously described as between the key file details and the environment and application are made. If the comparison is positive, then the protected application is allowed to run normally. If the comparison proves negative, then the protected application is permitted to run by the registration routine in demonstration mode only. If a duplicate key file is found not to exist at all and the internal key file, if present, brings a negative result, then the protected application is allowed to run in demonstration mode only.

This arrangement provides improved durability for the registration routine in the sense that it is less likely that the protected application will be caused to run in demonstration mode for incorrect reasons.

Third Embodiment

Tracking System

With reference to FIG. **4**, a modified form of the dialogue box C of FIG. **2***b* is shown which includes provision for entry of "your user number" in box **21**.

At the time a prospective new user enters his/her details into the other boxes comprising the dialogue box C, there is an option for the user to enter a user number into box **21**. The user number is provided by the registration authority **16** as a number unique to that particular registered user. If the box **21** has the user number details inserted into it, then the registration routine, When the next copy of the protected application is made, will transfer the user number details from box **21** to the "last user number" box **22**. A similar transfer will take place when next a copy is made of the protected application if and only if the person wishing to register the next copy enters their user number details in box **21**. If they do not, then the last user number details in box **22** remain as before. In this manner, a tracking system is available to the registration authority in the form of a tree where any given copy is identified by its ancestry based on current and previous user number as entered into boxes **21** and **22**.

Self-Serialization

In a particular embodiment, a process termed "self-serialization" can be utilized to produce the serial number **50** which is displayable to the user/licensee as illustrated in FIG. **4**.

The serial number **50** is disguised by use of a random or pseudorandom number input to the algorithm which generates the serial number at the time of first boot of the software as part of the initial registration procedure. For example, the serial number, when generated by the self-serialization process, can be generated by a random number routine forming part of the registration software or it can be generated by the registration software with reference to data which is available in a widely varying fashion on the platform on which the software is located—for example, a time reference on the platform. The serial number **50** generated by the self-serialization process can be a required input to the registration algorithm from which the registration number is generated. Clearly, the serial number **50**, as determined and displayed to the user, will then be required to be communicated to the registration authority for input to the registration authority's registration number generating algorithm.

It will be observed that a serial number **50** generated in this manner is likely to be displayed as a different number on each platform on which the software to be protected is to be run and comprises a randomized input to the registration algorithm which is determined and determinable only at the time of registration.

Fifth Embodiment

With reference to FIG. **5**, there is shown in schematic form a microprocessor **30** adapted to operate under an operating system or upon a platform **31** such as, for example, MicroSoft DOS or Macintosh System **7**. The platform **31** allows relatively high level commands to be used to cause the microprocessor **30** to interact with input/output devices such as keyboard **32**, monitor **33**, loudspeaker **34**, memory **35** and magnetic or CD ROM disk **36**.

By way of example a word processing program comprising a length of code or digital data **37** has been copied onto disk **36**.

The digital data **37** includes registration code portion **38** and use code portion **39**.

The digital data **37** is arranged in such a way that when microprocessor **30** seeks to first execute the digital data **37** by way of operating system or platform **31** the digital data comprising the registration code portion **38** is caused to execute first in a manner previously described in reference to the first embodiment of the invention. The execution of the digital data comprising the registration code portion **38** in conjunction with the operating system or platform **31** comprises a mode switcher which will permit the microprocessor **30** to execute the use code portion **39** of digital data **37** only in a demonstration mode unless and until registration involving reference to an external registration authority is first completed successfully. This registration procedure is as previously described with reference to the first embodiment.

The digital data **37** can comprise, for example, a word processing program such as Wordperfect 6.0 available from Wordperfect Corporation. The registration code portion **38** is integral with the digital data **37** comprising the word processing program. The registration code portion **38** includes the algorithm for calculation of the registration number as previously described in respect of other embodiments of the invention.

It will be appreciated that the registration code portion **38** effectively forms simply a part of the software or digital data **37** to be protected/registered and that the digital data **37** will be or can be identical for all copies of the word processing program produced. The registration code portion **38** allows a unique link to be made between the digital data **37** and an individual authorized or licensed to use the digital data **37** by way of initial execution of a copy of the digital data comprising registration code portion **38**.

With reference to FIGS. **6** and **7**, a specific realization of the fifth embodiment will be described.

With particular reference to FIG. **6**, a decoder **51** is interposed in the datapath from the CD in CD player **52** and a digital-to-analog converter **53**. The digital-to-analog converter **53** is the device by which digitally encoded musical or video information residing on CD ROM **54** is converted to analog form suitable for playback on current mass produced television sets (video) or hi-fi sets (audio).

The decoder **51** comprises part of the platform upon which the digital data **37** is executed and includes means to interpret the code portion **38** of the digital data **37** whereby the registration system is implemented such that the digital data **37** and, more particularly, the use code portion **39** of that digital data **37** can be executed on the platform in a use mode only if the registration procedure to which reference has been made in respect of previous embodiments has been performed.

The registration code portion **38** can include a preview or demonstration related to a subset of the balance of the digital data on the CD **54** which can be executed by the platform without license.

The decoder **61** includes LCD display **55** and keypad **56** whereby the licensee can enter information via keypad **56** and receive information via the LCD display **55** for the purpose of the registration procedure.

In addition a smart card (SRAM) **57** is receivable by the decoder **81** for the purpose of customizing or amending operation of the decoder **51**.

With reference to FIG. **7**, the registration procedure following insertion of CD **54** into CD player **52** is as follows. The user operates the play control and decoder **51** reads from CD **54** code portion **38** of digital data **37** located thereon and executes this code so as to determine whether the digital data is already licensed for the platform. If not, a demonstration is communicated via digital-to-analog converter **63** whilst the user determines whether to register as a licensee of the digital data **37** in the manner indicated in the flowchart of FIG. **7**.

Sixth Embodiment

With reference to FIG. **8**, there is shown a block diagram of a system according to a further embodiment of the invention which is to be read in the context of the earlier generalized description in respect of FIG. **1**.

The system illustrated in FIG. **8** operates in the manner generally described in respect of previous embodiments and as generally outlined in the diagram. In the context of the block C illustrated in FIG. **4**, and with reference to FIG. **9**, the algorithm, which generates the unique user identification and which is resident both as the registration code portion **38** in digital data **37** integrally bound to use code portion **39** for execution on local platform **31** and also as remote algorithm **61**, is attached to registration database program **62** for execution on the remote platform **63**.

The algorithm, in this embodiment, combines by addition the serial number **50** with the software product name **64** and customer information **65** and previous user identification **22** to provide registration number **66**.

As discussed earlier, all of the items to be summed, namely items **50**, **64**, **65** and **22** must be communicated to the remote licensee unique ID generator **67** by the intending licensee whereby algorithm **51** causes the production of a registration number **66** which matches identically with the locally produced registration number. When mode switcher **68** verifies the match, then the mode switcher **68** allows execution on platform **31** of the full user program **39**.

Prior to allowing execution of the full program, mode switcher **68** will also check whether platform ID **69** has changed as provided to it by platform unique ID generator **70**.

In this embodiment, serial number **50** is comprised of two components, namely system information **71** and a variable key portion **72**. The variable key portion **72** provides the characteristic of self serialization described earlier in the specification and, in this embodiment, is generated at the time of registration on platform **31** by reference to a variable platform parameter, in this case reference to system time information, although other parameters which are variable can be utilized in other embodiments.

System information **71** can include information which identifies the hardware comprising the platform **31** on which the user program **39** is to be executed such as, for example, CPU number (where available), or unique parameters associated with the firmware in use. The system information, optionally, can further include system configuration information such as amount of memory, type of processor etc.

It will be noted, therefore, that serial number **50** will appear to an intending licensee when it appears on screen as per box C in FIG. **4** as an apparently random variable having no obvious link to the platform **31** or the user program **39**.

However, when the serial number **50** is communicated to the remote licensee unique ID generator **67** a secondary algorithm complementary to the algorithm which generated the serial number including variable key portion **72** and system information **71** is able to "decode" or otherwise strip away the variable key portion **72** so as to make use of the system information **71** if allowable and desirable in the circumstances.

Whether the system information **71** is utilized or not, the serial number **50** generated in this manner provides an input to the algorithm which generates registration number **66** which presents as an apparently variable parameter thereby rendering "cracking" of the software registration system more difficult and unlikely.

Seventh Embodiment

The schematic diagram of FIG. **10** illustrates a substantially hardware implementation of the invention applicable, for example, for implementation of the CD arrangement of FIG. **6** or the more generalized arrangement of FIGS. **8** and **9**.

In this embodiment, a prospective user **80** of digital code **81** on media **82** by its execution on platform **83** firstly inserts the media **82** into an appropriate digital code reading device within platform **83** (e.g., a floppy disk drive or a CD ROM drive).

Customer information C is provided by user **80** both direct to local encoder/decoder **84** and also to local adder or summer **85**.

Additionally, product information P derived from media **82** (typically via platform **83**) or else via the intermediary of the user (signified by the small man symbol) is provided to encoder/decoder **84** and to summer **85**.

Finally, a serial number S derived from platform **83** is supplied either directly or via the intermediary of user **80** to encoder/decoder **84** and to summer **85**.

Summer **85** acts as a local licensee unique ID generating means by combining, by addition, customer information C, product information P and serial number S in order to provide a local licensee unique ID here designated Y.

Encoder/decoder **84** transmits the serial number S, the customer information C and the product information P via modems **86**, **87** over the public switched telephone network to a remote encoder/decoder **88** which, in turn, supplies signals S, C and P to the inputs of remote summer **89**. Remote summer **89** combines these signals by addition (thereby acting as a remote license unique ID generating means) so as to provide a summed output, here termed X, which represents a licensee unique ID or enabling key which should match identically with the local licensee unique ID or registration key or registration number Y if inputs S, C and P to summers **85** and **89** are identical.

The licensee unique ID termed X is transmitted back via encoder/decoders and modems **84**, **86**, **87**, **88** to comparator **90** which outputs a high signal if X equals Y. This condition corresponds to the local licensee unique ID matching with the licensee unique ID generated at the remote location by the remote licensee unique ID generating means generally comprising summer **89**.

Digital code **81** on media **82** comprises code identified as a demonstration portion D together with code identified as a use portion U. There may be other kinds of code designated O as well.

Code **81** is executed on platform **83** (for example, a microprocessor or a substantially hardware based, dedicated playback device such as a CD drive) with the code being passed through a mode switcher comprising first gate **91** and second gate **92** together with relay **93**.

First gate **91** energizes relay **93** so as to permit execution of code of type D but not code of any other type such as of type U.

Second gate **92** permits execution of any kind of code by closure of relay **93** provided only that the output of comparator **90** is high (which is to say that X equals Y or that the local licensee unique ID matches with the licensee unique ID generated by the remote licensee unique ID generating means comprising summer **89**).

Comparator **90** together with gates **91**, **92** and relay **93** comprise one particular form of mode switcher or switching platform **83** of various kinds of code such as the code of types D and U.

Industrial Applicability

The aforementioned may be applied either in dedicated electronic hardware or by means of more generalized digital computation devices such as microprocessors and the like whereby digital code or software (which may incorporate at least part of the code which, when executed, acts as a licensee unique ID generator) is fully enabled only after following a specified licensing procedure.

The above describes only some embodiments of the present invention and modifications, obvious to those skilled in the art, can be made thereto without departing from the scope and spirit of the present invention.

What is claimed is:

1. A registration system for licensing execution of digital data in a use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID first generated by said local licensee unique ID generating means has matched a licensee unique ID subsequently generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

2. The system of claim **1**, wherein said local licensee unique ID generating means generates said local licensee unique ID by execution of a registration algorithm which combines information in accordance with said algorithm, said information uniquely descriptive of an intending licensee of said digital data to be executed in said use mode.

3. The system of claim **2**, wherein said mode switching means permits operation of said digital data in said use mode in subsequent execution of said digital data only if said licensee unique ID generated by said local licensee unique ID generating means has not changed.

4. The system of claim **3**, wherein said local licensee unique ID generating means comprises part of said digital data when executed on said platform.

5. The system of claim **4**, wherein said mode switching means comprises part of said digital data when executed on said platform.

6. The system of claim **5**, wherein the information utilized by said local licensee unique ID generating means to produce said licensee unique ID comprises prospective licensee details including at least one of payment details, contact details and name.

7. The system of claim **1**, said system further including platform unique ID generating means, wherein said mode switching means will permit said digital data to run in said use mode in subsequent execution of said digital data on said platform only if said platform unique ID has not changed.

8. The system of claim **7**, wherein said platform unique ID generating means comprises part of said digital data when executed on said platform.

9. The system of claim **8**, wherein said platform unique ID generating means utilizes hard disc or other platform information to determine said platform unique ID.

10. The system of claim **1**, wherein said platform comprises a computer operating system environment.

11. The system of claim **10**, wherein said digital data comprises a software program adapted to run under said operating system environment.

12. A registration system attachable to software to be protected, said registration system generating a security key from information input to said software which uniquely identifies an intended registered user of said software on a computer on which said software is to be installed; and wherein said registration system is replicated at a registration authority and used for the purposes of checking by the registration authority that the information unique to the user is correctly entered at the time that the security key is generated by the registration system.

13. The registration system of claim **12**, wherein said security key is generated by a registration number algorithm.

14. The registration system of claim **13**, wherein said registration number algorithm combines information entered by a prospective registered user unique to that user with a serial number generated from information provided by the environment in which the software to be protected is to run.

15. The registration system of claim **12**, wherein said registration system checks at the time of boot of said software as to whether it is a first boot of the software to be protected or a subsequent boot, and, if a subsequent boot is detected, then environment and user details are compared to determine whether the program reverts to a demonstration mode and a new user registration procedure is to commence or a full version run.

16. The registration system of claim **15**, wherein said environment details comprise at least one element which is not user-configurable on the platform.

17. A method of control of distribution of software, said method comprising providing mode-switching means associated with said software adapted to switch said software between a fully enabled mode and a partly enabled or demonstration mode, said method further comprising providing registration key generating means adapted to generate a registration key which is a function of information unique to an intending user of the software; said mode-switching means switching said software into fully enabled mode only if an enabling key provided to said mode-switching means by said intending user at the time of registration of said software has matched identically with said registration key; and wherein said enabling key is communicated to said intending user at the time of registration of said software; said enabling key generated by a third party means of operation of a duplicate copy of said registration key generating means.

18. The method of claim **17**, wherein said registration key is also a function of the environment in which said software is installed.

19. A remote registration station incorporating remote licensee unique ID generating means, said station forming part of a registration system for licensing execution of digital data in a use mode, said digital data executable on a platform, said system including local licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform

only if a licensee unique ID generated by said local licensee unique ID generating means has matched a licensee unique ID generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

20. A method of registration of digital data so as to enable execution of said digital data in a use mode, said method comprising an intending licensee operating a registration system for licensing execution of digital data in a use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID generated by said local licensee unique ID generating means has matched a licensee unique ID generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

* * * * *

EXHIBIT 96

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 97

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 98

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 99

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 100

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

EXHIBIT 101

THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY