(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 September 2008 (25.09.2008)

PCT

(10) International Publication Number
WO 2008/115221 A2

(51) **International Patent Classification:**
H04L 29/08 (2006.01) ***H04L 29/06*** (2006.01)

(21) **International Application Number:**
PCT/US2007/025656

(22) International Filing Date:
14 December 2007 (14.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/919,035 20 March 2007 (20.03.2007) US

(71) **Applicant** *(for all designated States except US):* **THOMSON LICENSING** [FR/FR]; 46, Quai A. Le Gallo, F-F92100 Boulogne-billancourt (FR).

(72) Inventors; and
(75) Inventors/Applicants *(for US only):* **LIANG, Chao** [CN/US]; 1572 - 78th Street, Brooklyn, New York 11228 (US). **GUO, Yang** [CN/US]; 1507 Pheasant Hollow Drive, Plainsboro, New Jersey 08540 (US). **LIU, Yong** [CN/US]; 325 Marine Aenue, Apt. B3, Brooklyn, New York 11209 (US).

(74) Agents: LAKS, Joseph, J. et al.; c/o Thomson Licensing LLC, Two Independence Way Suite 200, Princeton, New Jersey 08540 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, **BR,** BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, **HR,** HU, **ID,** IL, IN, IS, **JP,** KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

(54) Title: HIERARCHICALLY CLUSTERED P2P STREAMING SYSTEM

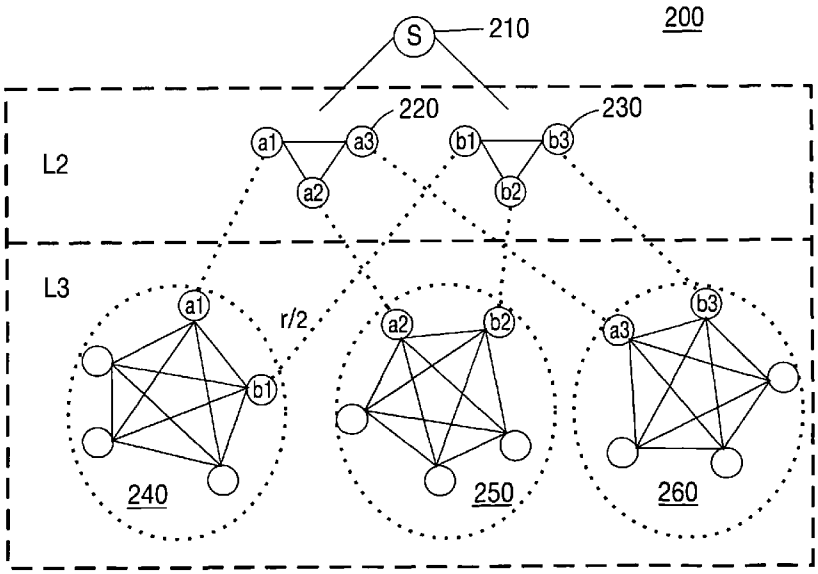

FIG. 2

(57) Abstract: A HCPS (hierarchically clustered P2P streaming system) comprising peers grouped into clusters and hierarchies. The HCPS actively balances the uploading capabilities among clusters and executes an optimal scheduling algorithm within each cluster to ensure that system resources are optimally utilized. The HCPS comprises an architecture which can be used in practical applications, yet can achieve the streaming rate close to the theoretical upper bound.

WO 2008/115221 A2

HIERARCHICALLY CLUSTERED P2P STREAMING SYSTEM

PRIORITY CLAIM

This application claims the benefit of United States Provisional Patent Application No. 60/919,035, filed March 20, 2007, entitled "HIERARCHICALLY CLUSTERED P2P STREAMING SYSTEM" which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

This invention relates to a method for creating a network from technical devices, such as digital electronic consumer devices and/or computers.

In computer technology it is well known to build up a network of connected devices for exchanging data and sharing hardware resources. The separate devices are commonly called nodes. At the time being, nodes are usually computers, but can be other technical devices, such as set top boxes, cellular telephones, mobile electronic devices or the like. The interconnections between the nodes are mainly electrically, optically or wireless radio connections. Networks can be classified as being based on either client-server or peer-to-peer (P2P) architectures. In P2P based networks a node is also referred to as a peer. While in client-server architectures each node is defined to be either client or server, there is no such differentiation in P2P networks. Instead, peers include both, server and client functionalities. P2P technology enables each node to be capable of providing services or resources to any other node in the network, or use services or resources provided by any other node in the network.

P2P networks are usually not restricted to any special applications or underlying network topologies, but can be understood as a set of nodes, or peers, which rely on certain sets of specific protocols. It is characteristic for a P2P network that the peers communicate directly with other peers, so that no central network organization is required. Most P2P networks support that peers can be connected to the network or disconnected from the network at any time.

The mentioned P2P protocols are required for basic network organization, such as for example, discovery of other connected peers, offering own services or resources to

other peers (advertising), understanding other peers' advertising messages, or allocating connection capacity for establishing certain connections to other peers. Also, there are protocols that enable a group of peers to cooperate, and thus form a peer-group. Such peer-groups are usually used for providing a common set of services within the peer group. Nevertheless, the purpose of a peer-group is not generally defined. A peer belonging to a peer-group normally has access to, and can be accessed from, all other connected peers of the same group. Additionally, each peer may be a member of further peer-groups. For adding or removing peers to or from a peer group, the user is always required to perform certain administrative activities.

Stochastic Fluid Theory for P2P Streaming Systems, (R. Kumar, Y. Liu, and K. Ross, in Proceedings of IEEE INFOCOM, 2007) teaches that the maximum video streaming rate in a P2P streaming system is determined by the video source server's capacity, the number of the peers in the system, and the aggregate uploading capacity of all peers. A perfect scheduling algorithm is also proposed to achieve the maximum streaming rate. In such a scheduling algorithm, each peer uploads the video content obtained directly from the server to all other peers in the system. To guarantee 100% uploading capacity utilization on all peers, different peers download different content from the server and the rate at which a peer downloads content from the server is proportional to its uploading capacity.

A perfect scheduling algorithm achieves the maximum streaming rate allowed by the system. Assuming n peers in the system, and peer i's upload capacity is u,, i = 1, 2, …, n. There is one source in the system with the upload capacity of u_s. Denoted by r^{max} the maximum streaming rate allowed by the system is:

$$r^{max} = \min\{w_{\Delta}, \frac{u_s + \sum_{i=1}^{n} u_i}{n}\} \quad (1)$$

The value of $(u_s + \sum_{i=1}^{n} u_i)/n$ is the average upload capacity per peer. Fig. 1 illustrates an exemplary system (100) according to the prior art demonstrating how the different portions of data are scheduled among three heterogeneous nodes with perfect scheduling algorithm. There are three peers (120, 130, 140) depicted in the system. Assuming the server (110) has capacity of 6 and the upload capacities of a (130), b (120) and c (130) are 2, 4 and

6 respectively, and that that all peers have enough downloading capacity, the maximum video rate that can be supported in the system is 6. To achieve that rate, the server divides video data into portions of 6 (151, 152, 153). A (130) is responsible for uploading 1 portion out of video data while b (140) and c (120) are responsible for upload 2 and 3 portions within each video data. This way, all peers (130, 140, 120) can download video at the maximum rate of 6. To implement such a perfect scheduling algorithm, each peer needs to maintain a connection and exchange video content with all other peers in the system. In addition, the server needs to split the video stream into multiple sub-streams with different rates, one for each peer. A real P2P streaming system can easily have a few thousand of peers. With current operating systems, it is unrealistic for a regular peer to maintain thousands of concurrent connections. It is also challenging for a server to partition a video stream into thousands of sub-streams in real time. Thus it is desirable to have a P2P streaming system which can achieve the streaming rate close to the theoretical upper bound, and the scheme is practical enough to use in practice.

SUMMARY OF THE INVENTION

In accordance with an aspect of the present invention, an apparatus and method for broadcasting data in a network of nodes is disclosed. The data may comprise video signals, audio signals, both audio and video signals, and forms of data, such as text, auxiliary data, or encoded information. According to an exemplary embodiment, the method for broadcasting a signal comprising the steps of receiving a plurality of requests for said signal from a plurality of peers, determining the number of peers requesting data, organizing said plurality of peers into a plurality of subsets of peers, dividing said signal into a number of portions of said signal equal to the number of subsets of peers, designating a single peer within each subset of peers as a cluster head of said subset of peers, and transmitting a portion of said signal to said cluster head within each subset of peers.

In accordance with another aspect of the present invention, the method for broadcasting a signal further comprises the step of enabling each of the cluster heads to transmit their respective received portion of the signal to each of said other cluster heads, such that each cluster head receives each of the portions of the signal from each of the other cluster heads,

thereby enabling each cluster head to generate a recombined signal representative of the signal.

In accordance with another aspect of the present invention, the apparatus comprises an interface for receiving requests for data from a plurality of nodes, a processor for determining the number of nodes requesting data, for organizing said plurality of nodes into a plurality of subsets of nodes, for dividing the data into a number of portions equal to the number of subsets of nodes, and a transmitter for sending a portion of the data to a cluster head of each subset of nodes.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an exemplary system (100) according to the prior art depicting a P2P architecture for using a perfect scheduling algorithm.

FIG. 2 illustrates a Hierarchically Clustered P2P Streaming System according to the present invention.

FIG. 3 illustrates an exemplary embodiment teaching a method of handling a new peer join according to the present invention.

Fig. 4 illustrates an exemplary embodiment teaching a method of handling a peer departure according to the present invention.

FIG. 5 illustrates an exemplary embodiment of the first phase of cluster re-balancing according to the present invention.

FIG. 6 illustrates an exemplary embodiment of the second phase of re-balancing according to the present invention.

FIG. 7 illustrates an exemplary embodiment of the cluster merge process according to the present invention.

FIG. 8 illustrates an exemplary embodiment of the cluster split process according to the present invention.

FIG. 9 illustrates an exemplary embodiment of the architecture of bootstrap node according to the present invention.

FIG. 10 illustrates an exemplary embodiment of vertical expansion of an HCPS system according to the present invention.

DETAILED DESCRIPTION

Other than the inventive concept, the elements shown in the figures are well known and will not be described in detail. Also, familiarity with television broadcasting and receivers is assumed and is not described in detail herein. A hierarchically clustered P2P streaming scheme (HCPS) to address the scalability issue faced by perfect scheduling algorithm. Instead of forming a single, large mesh, HCPS groups the peers into clusters. The number of peers in a cluster is relatively small so that the perfect scheduling algorithm can be successfully applied at the cluster level. A peer may be a device, cell phone, television signal processing device, radio frequency receiver, computer, thin client, set top box, modem, remote device, bridge device or other like device arranged in a network like structure. One peer in a cluster is selected as the cluster head and works as the source for this cluster. The cluster heads receives the streaming content by joining an upper level cluster in the system hierarchy. A cluster head may be any peer in the cluster, selected upon any criteria determined by system designers. The cluster head may be a device, cell phone, television signal processing device, radio frequency receiver, computer, thin client, set top box, modem, remote device, bridge device or other like device.

Turning to FIG. 2, an exemplary embodiment of a simple example of the proposed system (200) according to the present invention is presented. In the system of FIG. 2, the peers (a1-a3, b1-b3) are organized into a two-level hierarchy (L2, L3). At the base level (L3), peers are grouped into small size subsets of peers, or clusters (240, 250, 260). The peers are fully connected within a cluster. The peer with the largest upload capacity (a1, b1, c1) is elected as the cluster head. At the top level (L2), all cluster heads and the video server forms two clusters (220, 230). Video server (210) distributes the content to all cluster heads (a1, b1, c1) using the perfect scheduling algorithm at the top level (L2). At the base level (L3), each cluster head (a1, b1, c1) acts as the video server in its cluster and distributes the downloaded video to other peers in the same cluster, again, using the perfect scheduling algorithm. The number of connections on each normal peer is bounded by the size of its cluster. Cluster heads (a1, b1, c1) additionally maintain connections in the upper level cluster (L2).

Assuming in an exemplary embodiment that a cluster size is bounded by N_{max}, and the source can support up to N_s top layer clusters. The two-layer HCPS system, as shown in FIG. 2, can accommodate up to $N_s(N_{max})^2$ peers. Assuming that N_s = 10 and N_{max} =20, HCPS can support up to 4,000 peer. The maximum number of connections a peer needs to maintain is 40 for cluster head and 20 for normal peer, which is quite manageable. More peers can be accommodated by adding more levels into the hierarchy (vertical expansion).

In designing a system according to the present invention, it is desirable that the the peers be clustered such that the supportable streaming rate can be maximized. The maximum streaming rate, r_{max}, for a given set of peers and the source is achieved using the perfect scheduling algorithm with fully connected mesh. The mesh constructed in HCPS is not fully connected, which may reduce the maximum supportable streaming rate. It is desirable to have peer clustering strategies that would allow HCPS to support the streaming rate close to r_{max}. Heuristic peer clustering strategy that allows HCPS to have good supportable streaming rate.

In order to formulate a desirable supportable streaming rate for a given HCPS mesh topology as an optimization problem, assuming C clusters, N peers, and one source in the HCPS mesh. Cluster c has Vc peers, c = 1,2,. . .C. Denote by u_i the peer i's upload capacity. A peer can participate in the HCPS mesh either as a normal peer, or as a cluster head in the upper layer cluster and a normal peer in the base layer cluster. Denote by u_{ic} the amount of upload capacity of peer i contributed to cluster c as a normal peer, and by h_{ic} the amount of upload capacity of peer / contributed to cluster c as a cluster head. We further denote by us the source upload capacity and by u_c^s the amount of source capacity used for a top layer cluster c. If r_c^{max} represents the maximum streaming rate for cluster c using perfect scheduling algorithm, the maximum supportable streaming rate for a given cluster-based HCPS mesh, r_{HCPS}, can be formulated as following optimization problem.

$$r_{CPS}^{HCPS} = \max_{\{u_{ic},h_{ic},u_c^s\}} \{\min[r_c^{max} \mid c = 1,2,...,C]\} \quad (2)$$

Subject to:

$$r_c^{\max} = \min\{\frac{\sum_{i=1}^{N}(u_{ic} + h_{ic})}{V_c}, \sum_{i=1}^{N} h_{ic} + u_c^s\} \quad (3)$$

$$\sum_{c=1}^{C} u_{ic} + h_{ic} \le u_i \quad (4)$$

$$\sum_{c=1}^{C} u_c^s \le u^s \quad (5)$$

where Eqn. (3) is true for all c, c=l,2,. ..,C, and Eqn. (4) is true all for i, i=l,2, ...,N. $u_{ic} = 0$ if peer *i* is not in cluster c; and $h_{ic} = 0$ if peer i is not cluster c's head.

The maximum supportable streaming rate for a given mesh topology is the streaming rate that can be supported by all clusters. Since the cluster head participates in both upper layer and lower layer clusters and the source's upload capacity is used by several top layer clusters, the supportable streaming rate for HCPS can be maximized by adjusting the allocation of clusters' upload capacity and source's upload capacity. (equation 2) The first term in Equation (3) represents the average upload capacity per peer; and the second term represents the cluster head's upload capacity (cluster head can be the source or a peer). Since the maximum value of streaming rate at cluster c, $r_c^{\max}$ is governed by the perfect scheduling algorithm, this leads to the Equation (3). Further, the amount of bandwidth of cluster heads allocated for the upper layer and low layer clusters must not surpass its total upload capacity. (equation 4) Finally, for the source, the total allocated upload capacity for all clusters must not surpass the source's total upload capacity. (equation 5).

It is desirable in a HCPS mesh topology to support a streaming rate close to the optimal rate r_{max}. Assuming there are 400 peers with one source node, the cluster size is 20, and the peers are grouped into 20 base layer clusters and one top layer cluster for cluster heads. The maximum supportable streaming rate for HCPS is computed according to the optimization problem as formulate in Equation (2).

According to Equation (2), the maximum supportable streaming rate, r_{HCPS}, takes the minimum cluster streaming rate among all clusters. The cluster streaming rate (equation 3) is the minimum of cluster average upload capacity and the cluster head's rate. The peers should

be divided into clusters with similar average upload capacity to avoid wasting resources. The discrepancy of individual clusters' average upload capacity per peer should be minimized. The cluster head's upload capacity should be as large as possible. The cluster head's capacity allocated for the base layer capacity has to be larger than the average upload capacity to avoid being the bottleneck. Furthermore, the cluster head also joins the upper layer cluster. Ideally, the cluster head's rate should be greater than or equal to $2r_{HCPS}$.

It is desirable that the number of peers in a cluster should be bounded from the above by a relative small number. The number of peers in a cluster determines the out-degree of peers, and a large size cluster prohibits a cluster from performing properly using perfect scheduling.

Due to the peer dynamics, i.e., the peers join and leave the system all the time, the HCPS mesh should be dynamically adjusted to have consistent high supportable streaming rate. HCPS system has a bootstrap node that has the knowledge of the entire network: peers in the system, their upload capacities, the mesh topology (the membership of a cluster and its cluster head), etc. Bootstrap node also runs an optimizer that solves the optimization problem as formulated in Equation (2). Meanwhile, the cluster head manages the cluster it belongs to. Its responsibility includes (i) executing perfect scheduling algorithm locally based on member peers' upload capacities and the amount of upload capacity allocated as cluster head; (ii) handling the local peer departures and crashes; (iii) handling the instructions from the bootstrap node. The instructions includes new peer joining the cluster, cluster merge, cluster split, and cluster head change; (iv) maintaining the communication with the bootstrap node. Periodically update the bootstrap node about the membership of the cluster.

Fig. 3 - Fig. 5 describe the handling of new peer join, peer departure, and cluster re-balancing operation, respectively.

Turning to Fig. 3, an exemplary embodiment teaching a method of handling a new peer join is shown (300). The new arrival contacts the bootstrap node first (305). Based on the peer's upload capacity, u, and the current supportable streaming rate, r^{HCPS}, the peer is classified (310). The peer is classified as HPeer(with large upload capacity) if $u \geq r^{HCPS} + \delta$, MPeer(with medium upload capacity) if $r^{cps} - \delta < u < r^{HCPS} + \delta$, and

LPeer(with small upload capacity) otherwise. All clusters whose number of peers is no greater than Nmax are eligible to accept the new peer, where *Nmax* is the maximum number of nodes allowed by a cluster.

If the upload capacity of the new peer, *u*, is greater than some eligible cluster heads' upload capacity by a margin (315), the peer is assigned to the cluster with the smallest cluster head upload capacity. The new peer is to replace the original cluster head, and the original head becomes the normal peer and stay in the cluster (320).

The bootstrap node redirects the new peer to the cluster head, and informs the cluster head that the new peer will replace it. The cluster head transfer the membership info to the new peer, and un-register itself from the upper layer cluster. The original cluster head becomes a normal peer from now on. The new peer assumes the role of cluster head by registering itself into the upper layer cluster, and inform peers in the cluster where it is the new head. The new head executes the perfect scheduling algorithm and runs the cluster. Note that the value of the margin is typically a configuration parameter. Since the overhead of cluster head change is heavy, typically the margin is set to be a relatively large value.

If the new peer does not replace any cluster head, it is assigned to a cluster according to the value of *u* and the average upload capacity in the cluster. In an exemplary embodiment of cluster assignment among peers, the peer is assigned to the cluster with the minimum average upload capacity (330) if the peer is HPeer (325); the peer is assigned to the cluster with the smallest number of peers (340) if it is MPeer (335); and the peer is assigned to the cluster with maximum average upload capacity (350) if it is LPeer (345). This distribution is to balance the upload resources among clusters. The new peer is redirected to the corresponding cluster head, and bootstrap node requests the cluster head to admit the new peer. The cluster head takes the new peer, and informs other peers in the cluster. The connections are set up between new peer and other peers, and the cluster head adjusts the transmitting rate by applying perfect scheduling. In case all clusters are full and cannot accept a new peer, the bootstrap node randomly selects one cluster and split it into two clusters.

Turning to Fig. 4, an exemplary embodiment teaching a method of handling a peer departure is shown (400). When a peer decides to depart (405), if the peer is a normal peer (407), it informs the cluster head of its departure (415). The cluster head take the peer off its cluster member list, and informs other peers its departure (450). The cluster head then re-calculates the sending rate to other peers based on perfect scheduling. The cluster head also informs the bootstrap node the peer's departure (460).

In case the departing peer is the cluster head (407), the peer informs the bootstrap node its departure (410). The bootstrap node selects one peer from existing peers in the cluster as the new cluster head (420). The bootstrap node then informs the selected node that it will be the cluster head (430). The new cluster head then takes over the cluster head's functionality (440). The new cluster head then takes the old cluster head off its cluster member list, and inform other peers its departure (450). The new cluster head then re-calculates the sending rate to other peers based on perfect scheduling. Cluster head also informs the bootstrap node the peer's departure (460).

If a peer is crashed, the handling is the same for the normal peer. The cluster head notices the peer's crash, and treats it the same way as a normal departure. If the cluster head crashes, a peer in the cluster can inform the bootstrap node. The bootstrap node selects the peer with largest upload capacity as the new cluster head. The bootstrap node behaves as the cluster head, and let the selected peer to replace itself to become the new cluster head.

The clusters may lose balance in terms of the number of peers and the amount of resources in a cluster as the result of peer dynamics. Turning to FIG. 5, the first phase of cluster re-balancing is shown. In HCPS, the bootstrap node periodically attempts to re-balance the clusters. At the end of an epoch, the bootstrap node first attempts to balance the cluster sizes. The clusters are sorted in the descending order of cluster size (510). If the gap between the clusters with the largest and the smallest number of peers (520) is greater than threshold = $max\{aN^{max}, \beta\overline{N}\}$, where $\overline{N}$ is the average cluster size (530), these two clusters will be merged and then split into two balanced clusters (540). The merge and split operation are described below, respectively. The above process continues until no clusters violate the condition (550). The process then commences the second phase of re-balancing (560) as further depicted in Fig. 6.

Turning now to FIG. 6, the second phase of re-balancing is shown (600). In the second phase of cluster re-balancing, the bootstrap node attempts to balance the resources. The clusters are sorted in the descending order of average upload capacity per peer (610). If the average upload capacity difference of the clusters with highest and lowest upload capacity (620) is greater than the threshold of $\theta\bar{u}$, where $\bar{u}$ is the system average upload capacity (630, these two clusters will be merged and then split into two balanced clusters (640).

Turning now to Fig. 7, the cluster merge process is shown (700). The bootstrap node informs the two cluster heads the decision of merge and indicate which cluster is to be merged (710). The merged cluster head un-registers itself from upper layer cluster (720), and sends the member list to the new cluster head (730). The new cluster head informs all peers in the new cluster the member list (740). Connections are set up among peers. The new cluster head also re-calculates the sending rate using perfect scheduling. The new cluster head executes the perfect scheduling within the enlarged cluster (750).

Turning now to Fig. 8, the cluster split process is shown (800). The goal of cluster split is to divide the peers into two clusters that have roughly the similar number of peers and similar average upload capacity per peer. The bootstrap node manages the entire process. The cluster head sorts the peers in the descending order of upload capacity. (810) The peer with the largest upload capacity becomes the cluster head of the emerging new cluster (820). In the following rounds, two peers at the top of sorted list are taken out of the list. (830) The one with the larger upload capacity is assigned to the cluster with smaller aggregate upload capacity (835) and the peer with smaller upload capacity is assigned to the cluster with larger aggregate upload capacity. The process continues until all peers are assigned (825). If there is only one peer in the last round (840), then the peer is assigned to the cluster with more aggregate upload capacity (845).

Once the cluster membership is decided, the new cluster head creates the new cluster (850). It registers itself into the upper layer cluster, and then broadcasts the member list to all peers (855). The cluster head computes the sending rate to each peer using the perfect

scheduling (860). The original cluster head also informs the peers remaining in the cluster of the new member list, and computes the new sending rate.

Dynamic peer management enables the HCPS to have self-adaptive ability to balance member clusters in order to achieve high streaming rate, which makes HCPS desirable in relation to other P2P streaming systems. Assuming, for example, that one node / needs to spend *ts* transmission delay to deliver one specified data segment to other peer, then by perfect scheduling algorithm the last peer in the top level to receive the data chunk will wait *tp+Nmax*ts* time, where *tp* is the propagation delay from the server to node *i*. Given the designed small number *Nmax* and conventional propagation and transmission delay, the delay in one cluster can be quite small. To the two-level HCPS system, the data chunk will encounter at most two such above processes to reach all the peers of the system, because the process for the normal peer in the base level fetching data from cluster head is similar as that of head fetching data from server.

Turning now to Fig. 9, the architecture of bootstrap node is shown (900). The bootstrap node has three key components (910-930). Communication interface component is responsible to handle the incoming/outgoing signaling between the peers and bootstrap node. Coordinating component executes the key algorithms. These algorithms include cluster re-balancing algorithm, peer join, peer departure, peer merge, peer split, etc. based on the description before. It uses the peer related info and cluster related information stored in the component 3 (930). It may also update the component 3 (930) based on the incoming signaling information. Component 3 (930) is a database that keeps track of peer related information and cluster related information. This information includes, but not limited to, the peer IP address, the peer upload capacity, if a peer is cluster head, which cluster the peer joins, etc.

The architecture of a peer node (not shown) is similar to that of a bootstrap node. A peer also has three components. A communication interface component is responsible to handle the communication signaling with other peers and bootstrap node. Coordinating component executes the key algorithms. If the peer is cluster head, it execute the perfect scheduling algorithm, and handles member peers arrival, departure/crash, cluster merge and

split with other cluster, etc. If the peer is a normal peer, it contacts the cluster head and bootstrap node to update its information. It also receives the instruction from cluster head and bootstrap node to replace other cluster head. Cluster related information database stores the information of the peers belonging to the same cluster.

Turning to FIG. 10, an exemplary embodiment of vertical expansion of an HCPS system is shown (1000). HCPS is capable of supporting a larger number of peers with two-level structure than other signal level P2P streaming systems. The vertically expanded system comprises a first video server (1010) and a first level of sources (1020, 1030) for the second level clusters (1051-1056). The number of peers can be further increased by adding additional levels, hereinafter called vertical expansion. A peer in the second-level cluster (1051-1056) can drive another two-level HCPS system as the video server. Of course the peer that heads the next two-level clusters (1040, 1050) need to have sufficient upload capacity to contribute in the current cluster, and contribute as the source to low-level clusters (1057-1062). With the vertical expansion, the number of peers increases exponentially as the level of the system increases.

CLAIMS

1. A method for processing a signal comprising the steps of:

receiving a plurality of requests for said signal from a plurality of devices;

organizing said plurality of devices into a plurality of subsets of devices;

dividing said signal into a number of portions of said signal equal to the number of subsets of devices;

designating a single device within each subset of devices as a cluster head of said subset of devices; and

providing a portion of said signal to said cluster head within each subset of devices.

2. The method broadcasting a signal according to claim 1 further comprising the step of:

distributing data to each of said cluster heads within each subset of devices, said data indicating the other devices within the subset of devices.

3. The method of broadcasting a signal according to claim 1 further comprising the steps of:

enabling each of the cluster heads to provide their respective received portion of the signal to each of said other cluster heads, such that each cluster head receives each of the portions of the signal from each of the other cluster heads, thereby enabling each cluster head to generate a recombined signal representative of the signal.

4. The method of broadcasting a signal according to claim 3 further comprising the steps of:

enabling each of the cluster heads within a subset of devices to divide said recombined signal representation of the signal in a number of parts and to provide one of those parts to each of the other devices within the subset of devices.

5. The method of broadcasting a signal according to claim 4 further comprising the steps of:

enabling each of the other devices within the subset of devices to provide their respective received part of the recombined representation of the signal to each of said other devices within the subset of devices, such that each other device within the subset of devices receives each of the parts of the recombined representation of the signal from each other device within the subset of devices, thereby enabling each other device within the subset of

devices to generate a dataset representative of the recombined signal representative of the signal.

6. The method of broadcasting a signal according to claim 1 further comprising the steps of:

receiving a request add an additional device to said plurality of devices;

classifying said additional device in response to the upload capacity of said additional device;

assigning said additional device to one of said plurality of subsets of devices in response to the average upload capacity of said one of said plurality of subsets of devices.

7. The method of broadcasting a signal according to claim 1 further comprising the steps of:

receiving a request remove a departing device to said plurality of devices;

determining whether said departing device is a cluster head within a subset of devices;

removing said device from said subset of devices; and

designating another device with said subset of devices in response to said departing device being a cluster head.

8. The method of broadcasting a signal according to claim 1 further comprising the steps of:

sorting the subsets of devices in response to the number of devices within each subset of devices;

combining the subset of devices with the largest number of devices and the subset of devices with the smallest number of devices into a combined subset of devices;

dividing said combined subset of devices into two balanced subsets of devices, wherein each balanced subset of devices has no more than one more device than the other balanced subset of devices.

9. The method of broadcasting a signal according to claim 1 further comprising the steps of:

sorting the subsets of devices in response to the average upload capacity of each subset of devices;

combining the subset of devices with the largest average upload capacity and the subset of devices with the smallest upload capacity into a combined subset of devices;

dividing said combined subset of devices into two balanced subsets of devices, wherein each balanced subset of devices has a similar average upload capacity.

10. An apparatus comprising:

A first interface for receiving requests for data from a plurality of devices;

a processor for organizing said plurality of devices into a plurality of subsets of devices, for dividing the data into a number of portions equal to the number of subsets of devices, and

a second interface for sending a portion of the data to a cluster head of each subset of devices.

11. The apparatus according to claim 10 said processor further operative to designate one of said plurality of devices in each of said plurality of subsets of devices as a cluster head in response to the upload capacity of the designated device.

12. The apparatus according to claim 10 said processor further operative to distribute data to each of said cluster heads within each subset of devices, said data indicating the other devices within the subset of devices.

13. The apparatus according to claim 10 said processor further operative to enable each of the cluster heads to provvide their respective received portion of the signal to each of said other cluster heads, such that each cluster head receives each of the portions of the signal from each of the other cluster heads, thereby enabling each cluster head to generate a recombined signal representative of the signal.

14. The apparatus according to claim 13, said processor further operative to enable each of the cluster heads within a subset of devices to divide said recombined signal representation of the signal in a number of parts and to provide one of those parts to each of the other devices within the subset of devices.

15. The apparatus according to claim 14, said processor further operative to enable each of the other devices within the subset of devices to provide their respective received part of the

recombined representation of the signal to each of said other devices within the subset of devices, such that each other device within the subset of devices receives each of the parts of the recombined representation of the signal from each other device within the subset of devices, thereby enabling each other device within the subset of devices to generate a dataset representative of the recombined signal representative of the signal.

16. The apparatus according to claim 10, said processor further operative to receive a request add an additional device to said plurality of devices, classify said additional device in response to the upload capacity of said additional device, and assign said additional device to one of said plurality of subsets of devices in response to the average upload capacity of said one of said plurality of subsets of devices.

17. The apparatus according to claim 10, said processor further operative to receive a request remove a departing device to said plurality of devices, determine whether said departing device is a cluster head within a subset of devices, remove said device from said subset of devices; and designate another device within said subset of devices in response to said departing device being a cluster head.

18. The apparatus according to claim 10, said processor further operative to sort the subsets of devices in response to the number of devices within each subset of devices, combine the subset of devices with the largest number of devices and the subset of devices with the smallest number of devices into a combined subset of devices, divide said combined subset of device into two balanced subsets of device, wherein each balanced subset of device has no more than one more device than the other balanced subset of devices.

19. The apparatus according to claim 10, said processor further operative to sort the subsets of devices in response to the average upload capacity of each subset of devices, combine the subset of devices with the largest average upload capacity and the subset of devices with the smallest upload capacity into a combined subset of devices, and divide said combined subset of devices into two balanced subsets of devices, wherein each balanced subset of devices has a similar average upload capacity.

20. A method of distributing a signal on a network comprising the steps of:

organizing a plurality of devices into a first cluster and a plurality of second clusters wherein the number of devices in the first cluster equals the number of second clusters,

dividing the signal into a plurality of portions equal to the number of devices in said first cluster;

distributing to each of said devices in said first cluster, a portion of a signal and information identifying one of said plurality of second clusters; such that:

each of said plurality of devices in said first cluster may distribute among the devices in said first cluster portions of signal, such that each device in said first cluster may generate a first recombined representation of said signal; and

each of said plurality of devices in said first cluster enabled to dividing the first recombined representation of said signal into a plurality of portions equal to the number of devices in one of the plurality of second clusters, said second cluster corresponding to said information identifying one of said plurality of second clusters; and

distributing to each of said devices in said second cluster, a portion of said first recombined representation of said signal such that each of said plurality of devices in said second cluster may distribute among the devices in said second cluster portions of said first recombined representation of said signal, such that each device in said second cluster may generate a second recombined representation of said signal

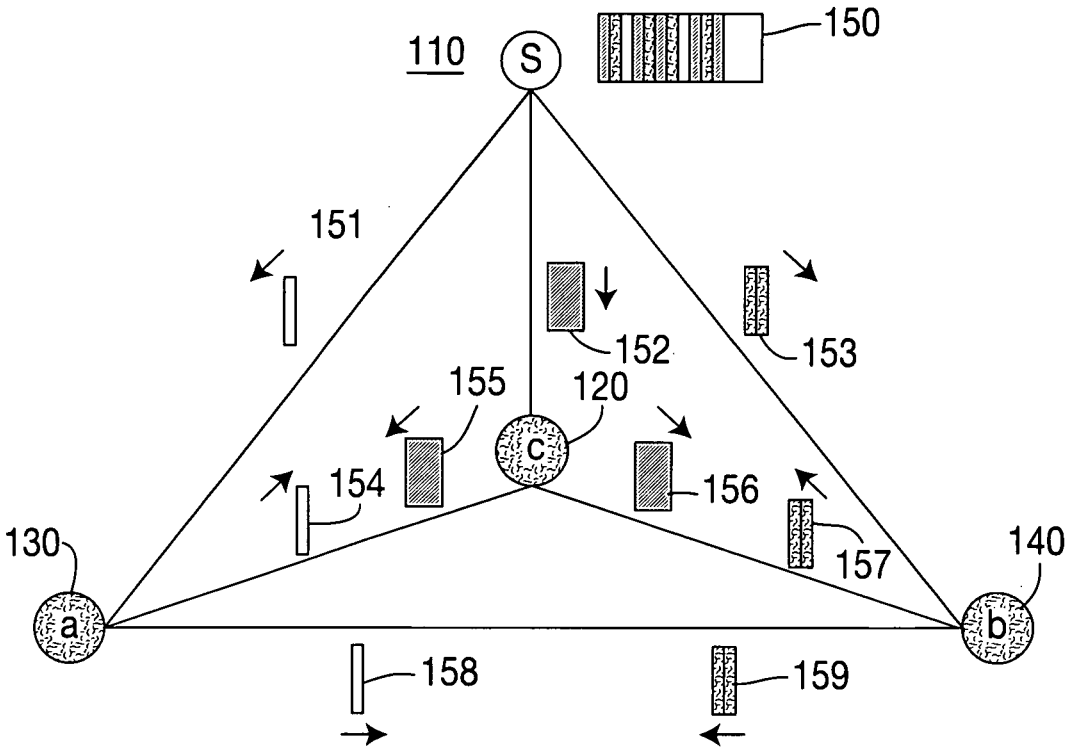

FIG. 1
PRIOR ART

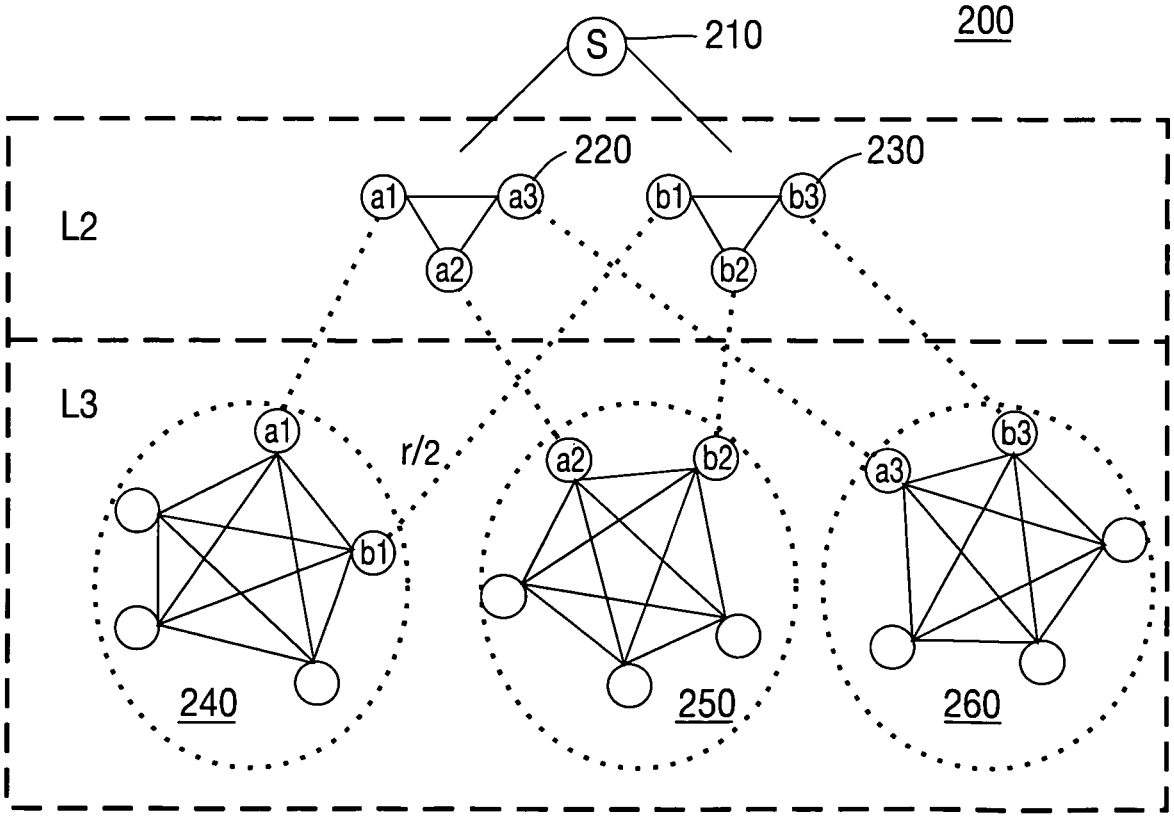

FIG. 2

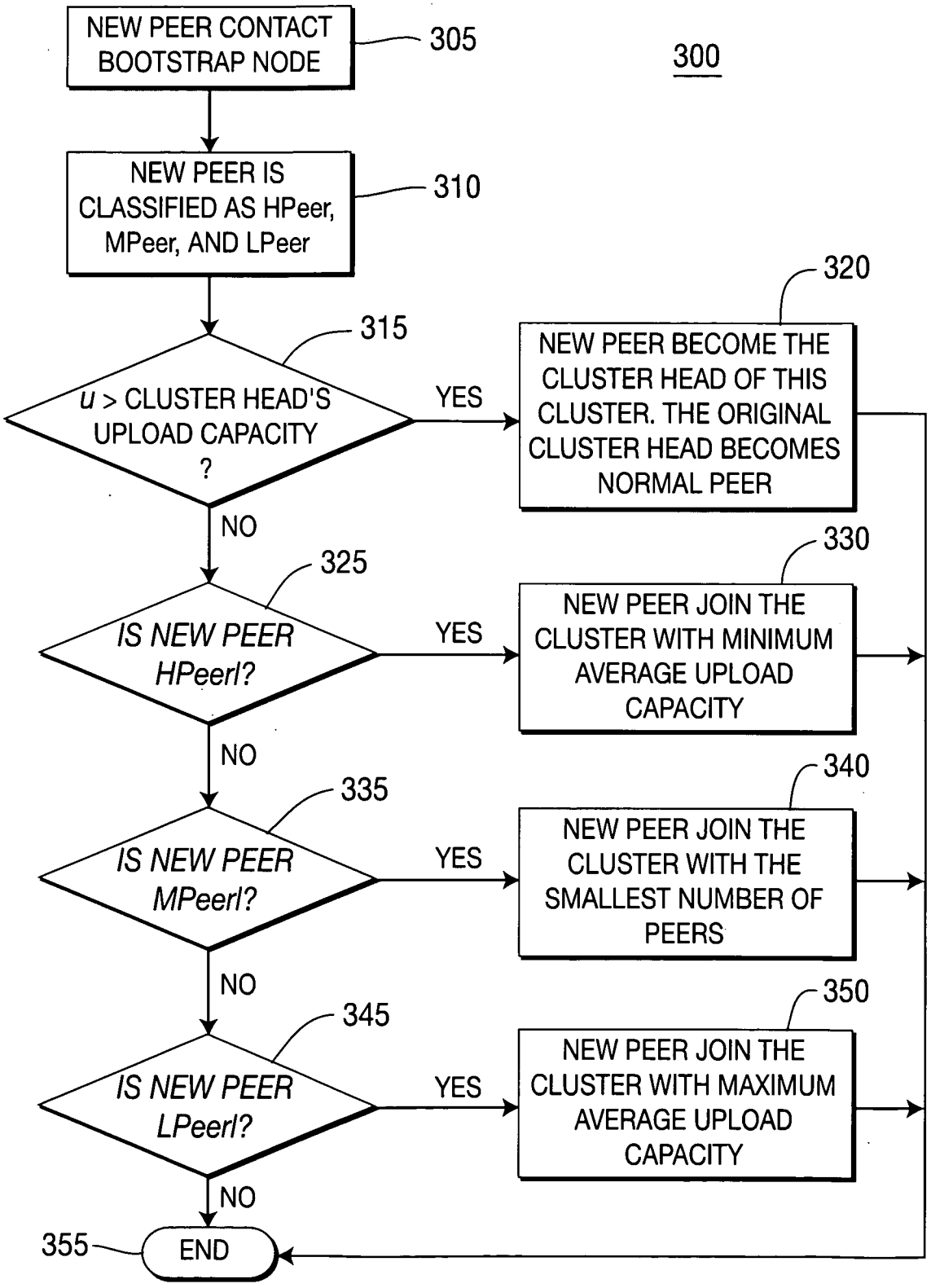
NEW PEER CONTACT BOOTSTRAP NODE
305
300
NEW PEER IS CLASSIFIED AS HPeer, MPeer, AND LPeer
310
315
u > CLUSTER HEAD'S UPLOAD CAPACITY ?
YES
320
NEW PEER BECOME THE CLUSTER HEAD OF THIS CLUSTER. THE ORIGINAL CLUSTER HEAD BECOMES NORMAL PEER
NO
325
IS NEW PEER HPeerl?
YES
330
NEW PEER JOIN THE CLUSTER WITH MINIMUM AVERAGE UPLOAD CAPACITY
NO
335
IS NEW PEER MPeerl?
YES
340
NEW PEER JOIN THE CLUSTER WITH THE SMALLEST NUMBER OF PEERS
NO
345
IS NEW PEER LPeerl?
YES
350
NEW PEER JOIN THE CLUSTER WITH MAXIMUM AVERAGE UPLOAD CAPACITY
NO
355
END

FIG. 3

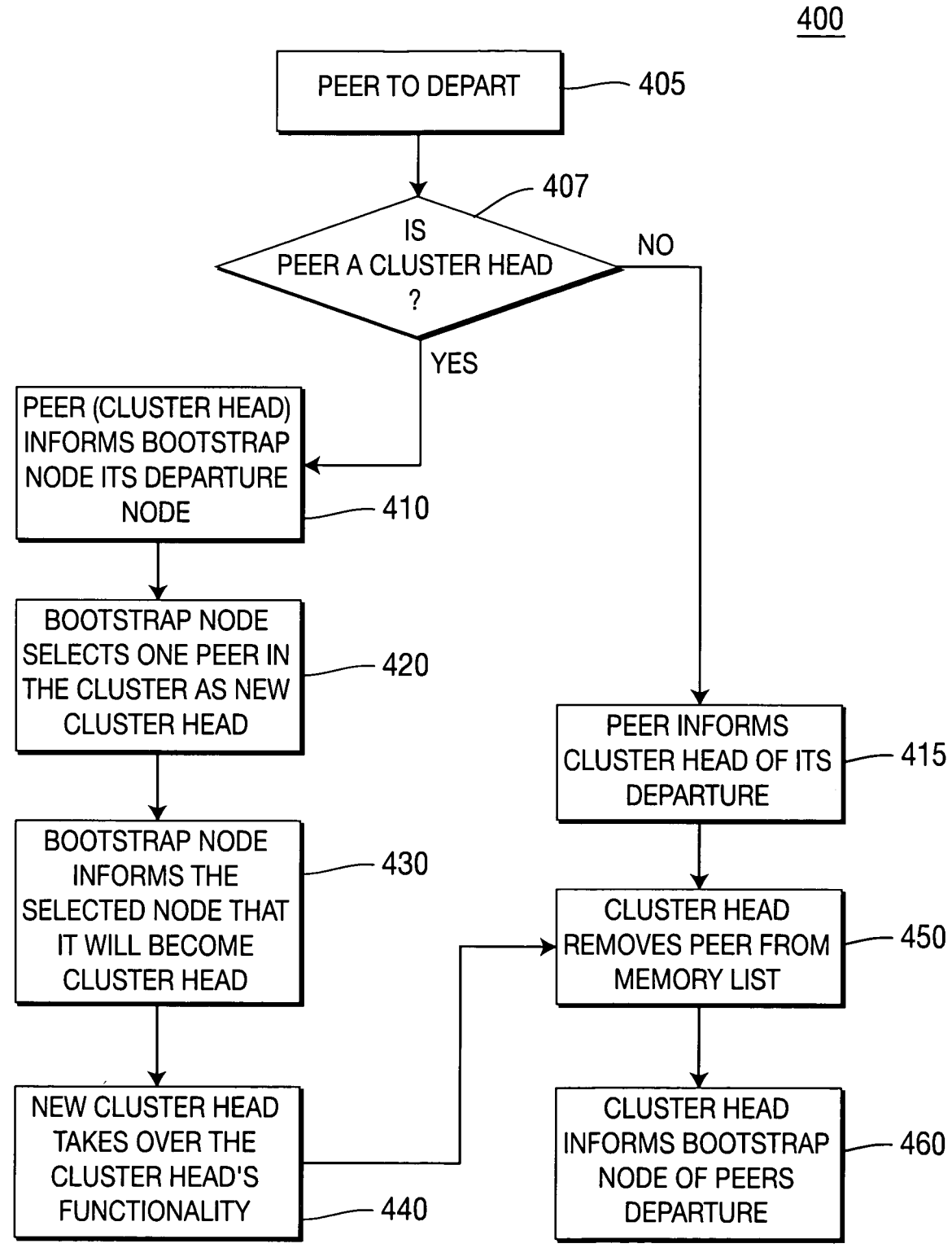
400
PEER TO DEPART
405
407
IS PEER A CLUSTER HEAD ?
NO
YES
PEER (CLUSTER HEAD) INFORMS BOOTSTRAP NODE ITS DEPARTURE NODE
410
BOOTSTRAP NODE SELECTS ONE PEER IN THE CLUSTER AS NEW CLUSTER HEAD
420
BOOTSTRAP NODE INFORMS THE SELECTED NODE THAT IT WILL BECOME CLUSTER HEAD
430
NEW CLUSTER HEAD TAKES OVER THE CLUSTER HEAD'S FUNCTIONALITY
440
PEER INFORMS CLUSTER HEAD OF ITS DEPARTURE
415
CLUSTER HEAD REMOVES PEER FROM MEMORY LIST
450
CLUSTER HEAD INFORMS BOOTSTRAP NODE OF PEERS DEPARTURE
460

FIG. 4

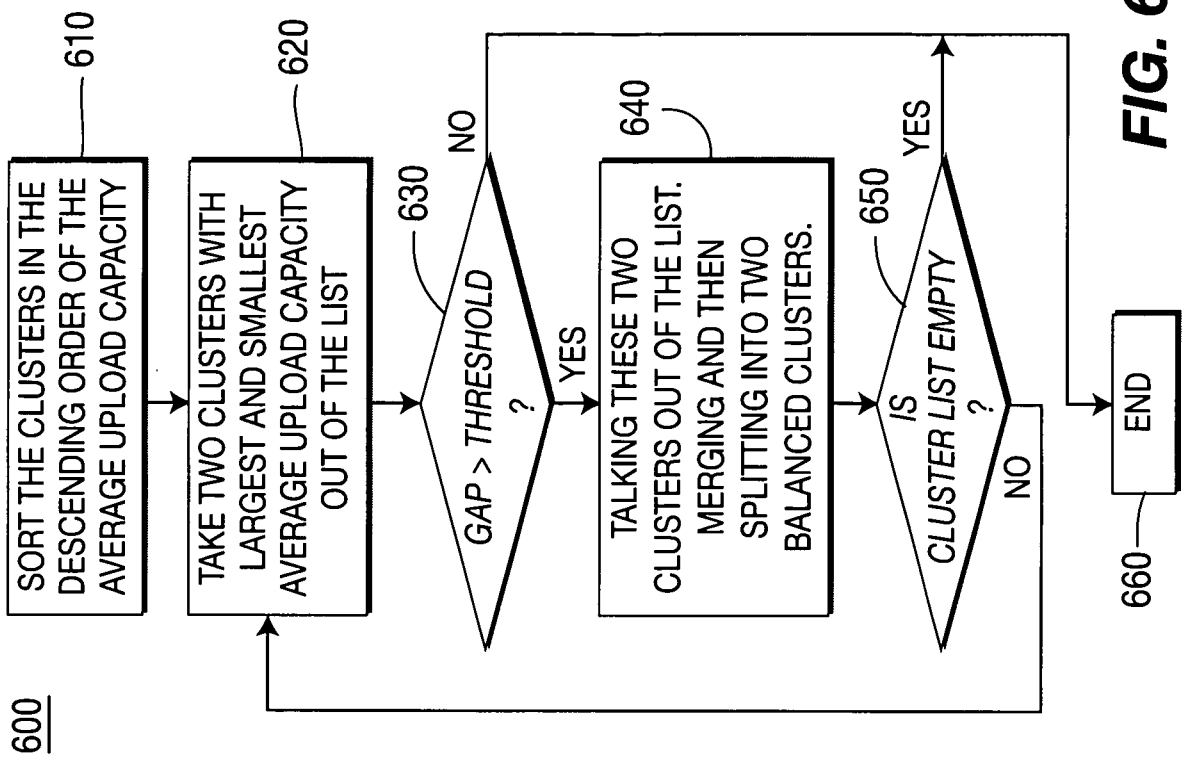
600
SORT THE CLUSTERS IN THE DESCENDING ORDER OF THE AVERAGE UPLOAD CAPACITY
610
TAKE TWO CLUSTERS WITH LARGEST AND SMALLEST AVERAGE UPLOAD CAPACITY OUT OF THE LIST
620
GAP > THRESHOLD ?
630
NO
YES
TALKING THESE TWO CLUSTERS OUT OF THE LIST. MERGING AND THEN SPLITTING INTO TWO BALANCED CLUSTERS.
640
IS CLUSTER LIST EMPTY ?
650
YES
NO
END
660

FIG. 6

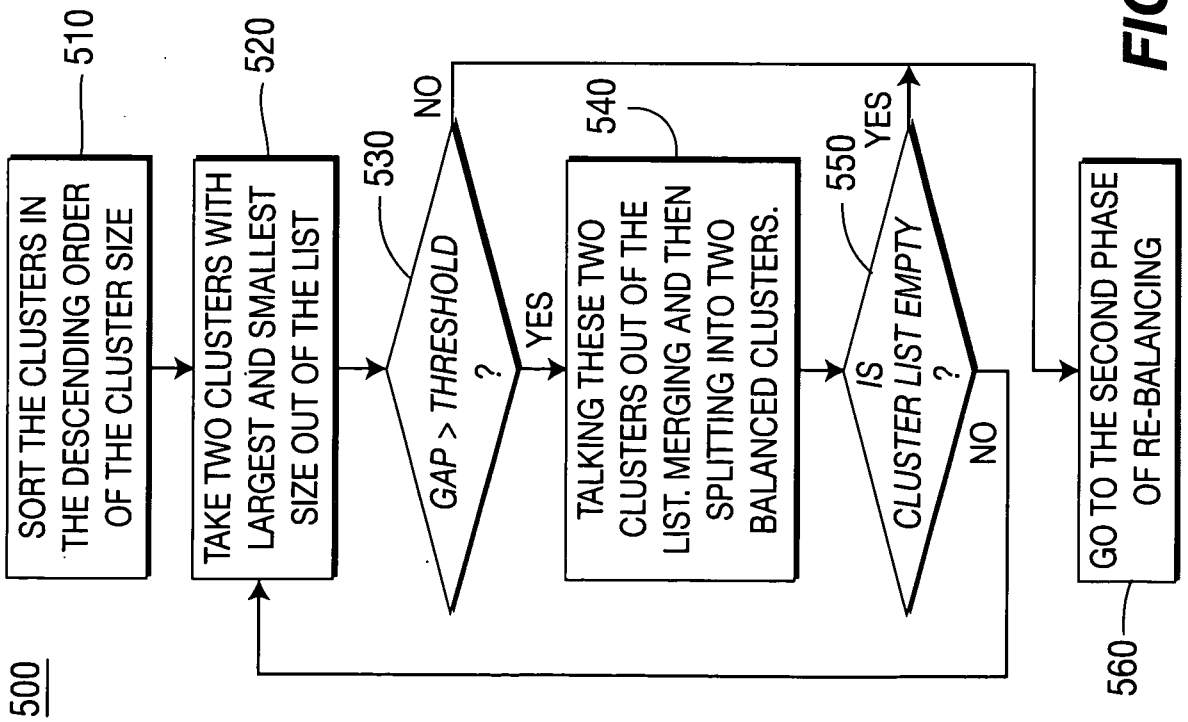
500
SORT THE CLUSTERS IN THE DESCENDING ORDER OF THE CLUSTER SIZE
510
TAKE TWO CLUSTERS WITH LARGEST AND SMALLEST SIZE OUT OF THE LIST
520
GAP > THRESHOLD ?
530
NO
YES
TALKING THESE TWO CLUSTERS OUT OF THE LIST. MERGING AND THEN SPLITTING INTO TWO BALANCED CLUSTERS.
540
IS CLUSTER LIST EMPTY ?
550
YES
NO
GO TO THE SECOND PHASE OF RE-BALANCING
560

FIG. 5

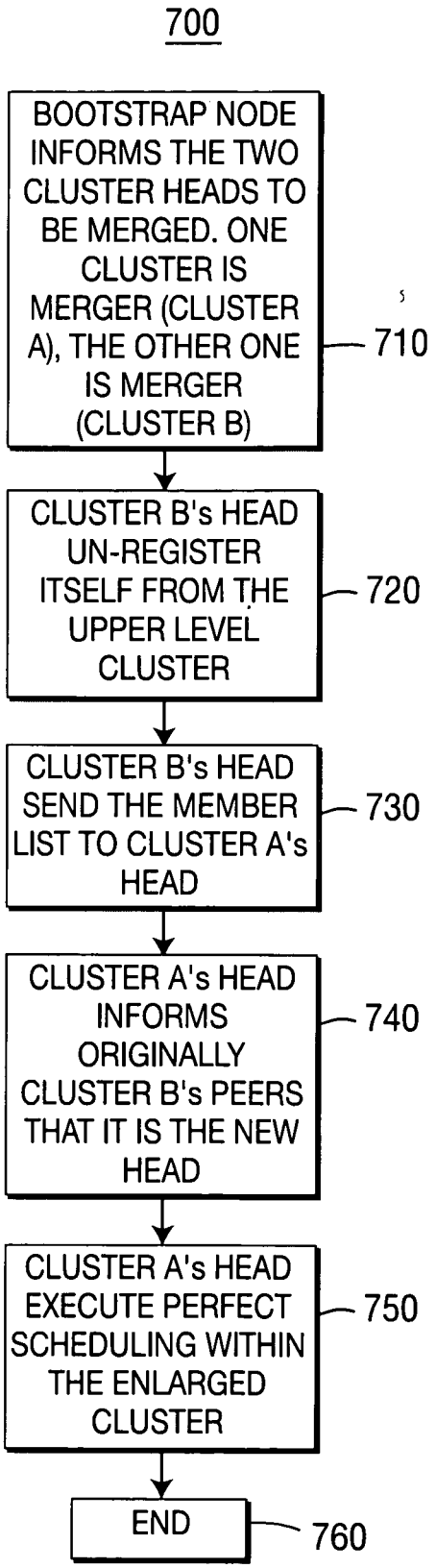
700
BOOTSTRAP NODE INFORMS THE TWO CLUSTER HEADS TO BE MERGED. ONE CLUSTER IS MERGER (CLUSTER A), THE OTHER ONE IS MERGER (CLUSTER B)
710
CLUSTER B's HEAD UN-REGISTER ITSELF FROM THE UPPER LEVEL CLUSTER
720
CLUSTER B's HEAD SEND THE MEMBER LIST TO CLUSTER A's HEAD
730
CLUSTER A's HEAD INFORMS ORIGINALLY CLUSTER B's PEERS THAT IT IS THE NEW HEAD
740
CLUSTER A's HEAD EXECUTE PERFECT SCHEDULING WITHIN THE ENLARGED CLUSTER
750
END
760

FIG. 7

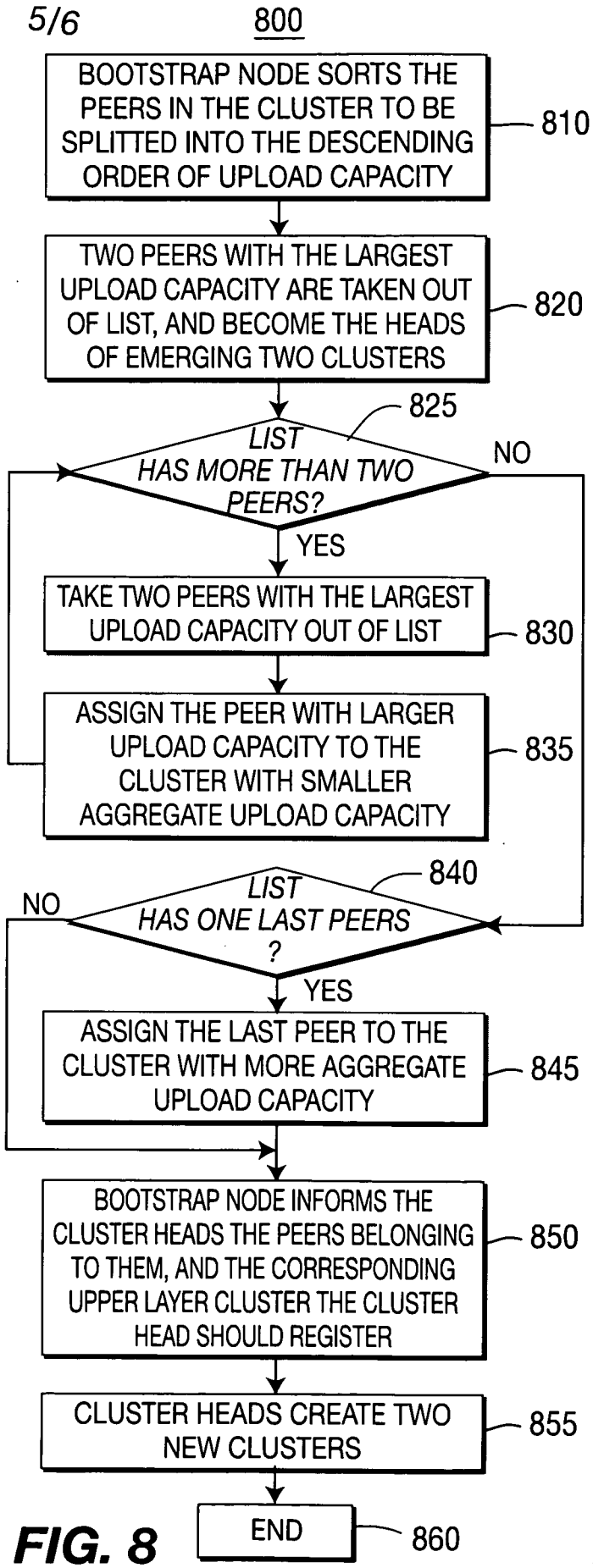
800
BOOTSTRAP NODE SORTS THE PEERS IN THE CLUSTER TO BE SPLITTED INTO THE DESCENDING ORDER OF UPLOAD CAPACITY
810
TWO PEERS WITH THE LARGEST UPLOAD CAPACITY ARE TAKEN OUT OF LIST, AND BECOME THE HEADS OF EMERGING TWO CLUSTERS
820
825
LIST HAS MORE THAN TWO PEERS?
NO
YES
TAKE TWO PEERS WITH THE LARGEST UPLOAD CAPACITY OUT OF LIST
830
ASSIGN THE PEER WITH LARGER UPLOAD CAPACITY TO THE CLUSTER WITH SMALLER AGGREGATE UPLOAD CAPACITY
835
840
LIST HAS ONE LAST PEERS ?
NO
YES
ASSIGN THE LAST PEER TO THE CLUSTER WITH MORE AGGREGATE UPLOAD CAPACITY
845
BOOTSTRAP NODE INFORMS THE CLUSTER HEADS THE PEERS BELONGING TO THEM, AND THE CORRESPONDING UPPER LAYER CLUSTER THE CLUSTER HEAD SHOULD REGISTER
850
CLUSTER HEADS CREATE TWO NEW CLUSTERS
855
END
860

FIG. 8

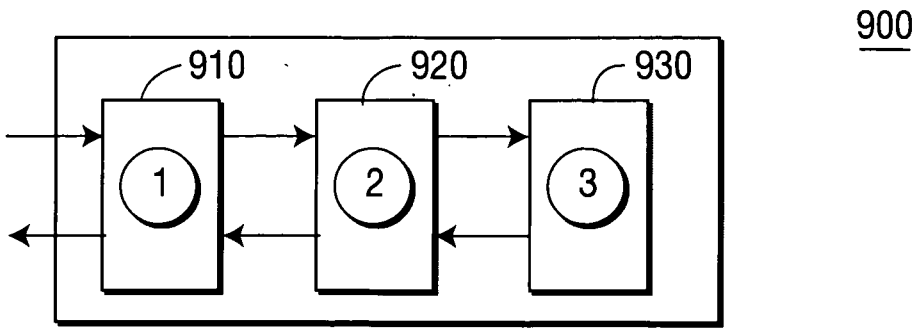
900
910
920
930
1
2
3

FIG. 9

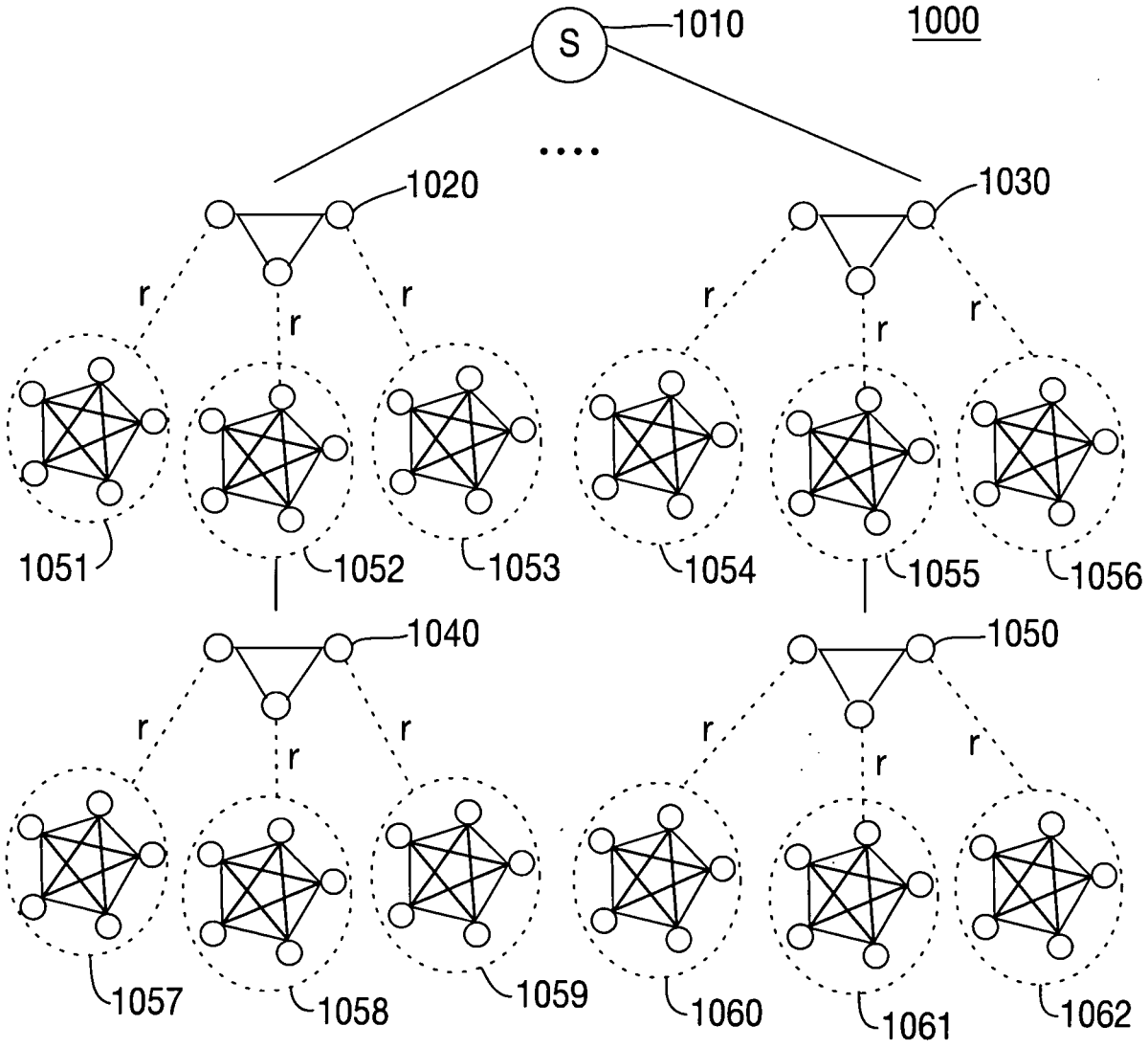
1000
S
1010
1020
1030
r
1051
1052
1053
1054
1055
1056
1040
1050
1057
1058
1059
1060
1061
1062

FIG. 10