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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 February 2015 (05.02.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/017145 A1

(51) International Patent Classification:
H04L 12/735 (2013.01) *H04L 12/707* (2013.01)
H04L 12/931 (2013.01) *H04L 12/933* (2013.01)

(21) International Application Number:
PCT/US2014/047280

(22) International Filing Date:
18 July 2014 (18.07.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/859,650 29 July 2013 (29.07.2013) US
14/226,288 26 March 2014 (26.03.2014) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, California 94065 (US).

(72) Inventors: **BOGDANSKI, Bartosz**; Hoff Terrasse 15 H0203, N-0275 Oslo (NO). **JOHNSEN, Bjørn Dag**; Vilberggrenda 9, N-0687 Oslo (NO).

(74) Agents: **MEYER, Sheldon, R.** et al.; Meyer IP Law Group, 410 Pacific Avenue, San Francisco, California 94133 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, 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, IR, IS, JP, KE, KG, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: SYSTEM AND METHOD FOR SUPPORTING MULTI-HOMED FAT-TREE ROUTING IN A MIDDLEWARE MACHINE ENVIRONMENT

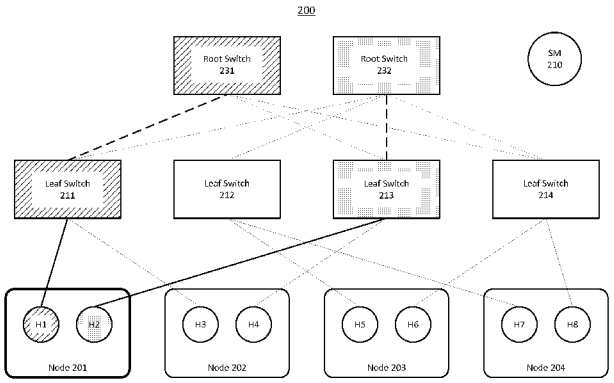

FIGURE 2

(57) Abstract: A system and method can support multi-homed routing in a network environment, which can be based on InfiniBand architecture using a fat-tree or a similar topology. The system can provide an end node that is associated with a switch port on a leaf switch in a network fabric. Then, the system can perform routing for each of a plurality of ports on the end node, and ensure that the plurality of ports on the end node take mutually independent paths.

WO 2015/017145 A1

SYSTEM AND METHOD FOR SUPPORTING MULTI-HOMED FAT-TREE ROUTING IN A MIDDLEWARE MACHINE ENVIRONMENT

Copyright Notice:

Field of Invention:

[0002] The present invention is generally related to computer systems, and is particularly related to a network environment.

Background:

[0003] The fat-tree topology is used for high performance computing (HPC) clusters, and for clusters based on InfiniBand (IB) technology. For example, the fat-tree topology is used in the fastest supercomputers, such as MilkyWay-2. Also, the fat-tree IB systems include large installations such as Stampede, TGCC Curie and SuperMUC.

[0004] These are the generally areas that embodiments of the invention are intended to address.

Summary:

[0005] Described herein are systems and methods that can support multi-homed routing in a network environment, which can be based on InfiniBand architecture using a fat-tree or a similar topology. The system can provide an end node that is associated with a switch port on a leaf switch in a network fabric. Then, the system can perform routing for each of a plurality of ports on the end node, and ensure that the plurality of ports on the end node take mutually independent paths.

Brief Description of the Figures:

[0006] **Figure 1** shows an illustration of supporting fat-tree routing in a network environment.

[0007] **Figure 2** shows an illustration of supporting multi-homed routing in a network environment, in accordance with an embodiment of the invention.

[0008] **Figure 3** shows an illustration of providing redundancy for supporting a fat-tree routing in a network environment, in accordance with an embodiment of the invention.

[0009] **Figure 4** illustrates an exemplary flow chart for supporting multi-homed routing in a network environment, in accordance with an embodiment of the invention.

[0010] **Figure 5** illustrates a block diagram of a computer system upon which an embodiment of the invention may be implemented.

[0011] **Figure 6** illustrates a block diagram of a system for implementing an embodiment of the invention.

Detailed Description:

[0012] The invention is illustrated, by way of example and not by way of limitation, in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" or "some" embodiment(s) in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0013] The description of the invention as following uses the Infiniband (IB) network as an example for a high performance network. It will be apparent to those skilled in the art that other types of high performance networks can be used without limitation. Also, the description of the invention as following uses the fat-tree topology as an example for a fabric topology. It will be apparent to those skilled in the art that other types of fabric topologies can be used without limitation.

[0014] Described herein are systems and methods that can support multi-homed routing in a network environment.

InfiniBand Architecture

[0015] The InfiniBand Architecture (IBA) supports a two-layer topological division. At the lower layer, IB networks are referred to as subnets, where a subnet can include a set of hosts interconnected using switches and point-to-point links. At the higher level, an IB fabric constitutes one or more subnets, which can be interconnected using routers.

[0016] Furthermore, hosts and switches within a subnet can be addressed using local identifiers (LIDs), and a single subnet can be limited to 49151 LIDs. Besides the LIDs, which are the local addresses that are valid only within a subnet, each IB device can have a 64-bit global unique identifier (GUID) that is burned into its non-volatile memory. A GUID can be used to form a global identifier (GID), which is an IB layer three (L3) address. A GID can be created by concatenating a 64-bit subnet identifier (ID) with the 64-bit GUID to form an IPv6-like 128-bit address. For example, different port GUIDs can be assigned to the ports connected to the IB fabric.

[0017] Additionally, a subnet manager (SM) can be responsible for performing routing table calculations in an IB fabric. Here, the routing of the IB network aims at obtaining full connectivity, deadlock freedom, and proper load balancing between all source and destination pairs in the local subnet.

[0018] The subnet manager can calculate the routing tables at network initialization time.

Furthermore, the routing tables can be updated whenever the topology changes, in order to ensure optimal performance. During normal operations, the subnet manager can perform periodic light sweeps of the network to check for topology changes. If a change is discovered during a light sweep or if a message (trap) signaling a network change is received by the subnet manager, the subnet manager can reconfigure the network according to the discovered changes.

[0019] For example, the subnet manager can reconfigure the network when the network topology changes, such as when a link goes down, when a device is added, or when a link is removed. The reconfiguration steps can include the steps performed during the network initialization. Furthermore, the reconfigurations can have a local scope that is limited to the subnets, in which the network changes occurred. Also, the segmenting of a large fabric with routers may limit the reconfiguration scope.

[0020] Additionally, an IB network, which is based on a lossless networking technology, may be prone to deadlocks under certain conditions. For example, deadlocks may occur in an IB network where network resources, such as buffers or channels, are shared and packet drops are not allowed. Here, a necessary condition for a deadlock to happen is the creation of a cyclic credit dependency, which means that a cyclic credit dependency can make the deadlock occurrence possible. On the other hand, this does not mean that there will always be a deadlock whenever a cyclic credit dependency is present.

Fat-tree Routing

[0021] The fat-tree topology can provide various benefits for supporting high performance interconnections. These benefits can include deadlock freedom, inherent fault-tolerance, and full bisection bandwidth. The deadlock freedom represents that the use of a tree structure makes it possible to route fat-trees without special considerations for deadlock avoidance. The inherent fault-tolerance represents that the existence of multiple paths between individual source destination pairs makes it easier to handle network faults. The full bisection bandwidth represents that the network can sustain full speed communication between the two halves of the network.

[0022] Furthermore, fat-tree routing algorithms can be used to support the efficient use of the underlying fat-tree topology. The following Algorithm 1 is an exemplary fat-tree routing algorithm.

Algorithm 1 *route_to_cns() function*

Require: Addressing is completed
Ensure: All *hca_ports* are routed
1: **for** *swleaf = 0* **to** *max_leaf_sw* **do**
2: **for** *swleaf.port = 0* **to** *max_ports* **do**

```
3:              hca_lid = swleaf.port-> remote_lid
4:              swleaf.routing table[hca_lid] = swleaf.port
5:              route_downgoing_by_going_ up()
6:      end for
7: end for
```

[0023] As shown in the above, the routing function, route_to_cns(), can iterate over an array of leaf switches (Lines 1-7). For each selected leaf switch, the routing function can route toward each end-node port that is connected to the selected leaf switch, e.g. in the port numbering sequence (Lines 2-6).

[0024] Furthermore, when routing an end-node port that is associated with a particular LIDs, the routing function can go up one level in the network topology to route the downgoing paths, and when routing each switch port, the routing function can go down to route the upgoing paths. This process can be repeated until the root switch level is reached. After that the paths towards all nodes are routed and inserted into the linear forwarding tables (LFTs) of all switches in the fabric.

[0025] For example, the route_downgoing_by_going_up() function (Line 5) can be a recurrence function that can balance the paths and call the route_upgoing_by going_down() function, which routes the upward paths in the fat-tree toward destination through the switch from which the route_downgoing_by_going_up() function was invoked.

[0026] There can be several potential drawbacks associated with the route_to_cns() function. First, the route_to_cns() function is oblivious and routes the end-ports without any consideration as to which end-node the end-ports belong. Second, the route_to_cns() function depends on the physical port number for routing.

[0027] **Figure 1** shows an illustration of supporting fat-tree routing in a network environment. As shown in Figure 1, one or more end nodes 101-104 can be connected to a network fabric 100. The network fabric 100 can be based on a fat-tree topology, which includes a plurality of leaf switches 111-114, and multiple spine switches or root switches 131-134. Additionally, the network fabric 100 can include one or more intermediate switches, such as switches 121-124.

[0028] Also as shown in Figure 1, each of the end nodes 101-104 can be a multi-homed node, i.e. a single node that is connected to two or more parts of the network fabric 100 through multiple ports. For example, the node 101 can include the ports H1 and H2, the node 102 can include the ports H3 and H4, the node 103 can include the ports H5 and H6, and the node 104 can include the ports H7 and H8.

[0029] Additionally, each switch can have multiple switch ports. For example, the root switch S1 131 can have the switch ports 1-2, the root switch S2 132 can have the switch ports 3-4, the root switch S3 133 can have the switch ports 5-6, and the root switch S4 134 can have the switch ports 7-8.

[0030] Using a fat-tree routing algorithm, such as Algorithm 1, which routes on leaf switch basis, there is no guarantee that independent routes will be assigned to different two-port nodes 101-104. For example, the ports H1, H2, H5 and H6 can be connected to port 1 on each switch (not shown), while the ports H3, H4, H7 and H8 are connected to port 2 on each switch (not shown). Here, after routing through four end-ports and traversing through the leaf switch 113 and the switch 123, the fat-tree routing algorithm may assign the two paths from the pair of end-ports, H1 and H2 on node 101, to the same leftmost root switch S1 131 (via switch ports 1-2 respectively). Similarly, other pair of end-ports, e.g. H3 and H4 on node 102, H5 and H6 on node 103, and H7 and H8 on node 104, may be routed through the same root switch (i.e. S2 132 – S4 134 respectively).

[0031] This can result in an undesirable behavior for a user. As shown in Figure 1, the end node 101 can have a single point of failure at the root switch S1 131, even though the end node 101 may have built-in physical fault-tolerance (i.e. two end-ports connected to different leaf switches 111 and 113) has. Additionally, depending on the physical cabling, similar problems may appear and the single point of failure may occur on other switches in the fat-tree topology.

[0032] Furthermore, within the network fabric 100, the traffic to different ports on the same node may be routed through a single link. Thus, this single link may represent both an additional single point of failure for the set of end ports, and a performance bottleneck (since the traffic targeting different end ports may effectively only be able to utilize the bandwidth of the single shared link).

Multi-homed Fat-tree Routing

[0033] In accordance with an embodiment of the invention, the system can provide independent routes for multi-homed nodes in fat-trees, so that a single point of failure may not lead to complete outage.

[0034] **Figure 2** shows an illustration of supporting multi-homed routing in a network environment, in accordance with an embodiment of the invention. As shown in Figure 2, a network environment can comprise a plurality of end nodes (e.g. nodes 201-204), each of which can include one or more ports (e.g. ports H1-H8). Additionally, the plurality of end nodes 201-204 can connect to a network fabric 200, which can be in a fat-tree topology. Also, the network fabric 200 can include a plurality of switches, e.g. the leaf switches 211-214 and the root switches, S1 231 and S2 232.

[0035] In accordance with an embodiment of the invention, a multi-homed fat-tree routing algorithm, such as the mFtree algorithm, can be used for performing the fat-tree routing, e.g. by a subnet manager (SM) 210. In the example as shown in Figure 2, the mFtree algorithm can identify the paths from the port H1 and the port H2 on node 201 may need to be routed in a mutually redundant way, since both ports locate on a single end node 1.

[0036] Furthermore, the mFtree algorithm can ensure that the paths are in fact redundant. For example, the path from the port H1 on node 201 can go through the leaf switch 211 and eventually leads to a root switch 231. As shown in Figure 2, the system can mark the switches in the path (as shown in dark shading). Then, the system can avoid using the marked switches for determining the path from the port H2 on node 201. Thus, the path from the port H2 on node 201 can go through a redundant path (e.g. via the leaf switch 213) and eventually leads to a different root switch 232 (as shown in light shading).

[0037] When the routing step for the node 201 is completed, the algorithm can mark the node as routed (as shown in bold line), so that the routing step is not repeated for the node 201 when the algorithm encounters another port of that node. Thus, the system can ensure that a single point of failure does not lead to a complete outage of a multi-port node.

[0038] Additionally, the fat-tree routing algorithm can provide improvements in performance, scalability, availability and predictability of InfiniBand (IB) fat-tree topologies.

[0039] The following Algorithm 2 is an exemplary multi-homed fat-tree routing algorithm.

Algorithm 2 *route_multihomed_cns() function*

Require: Addressing is completed
Ensure: All *hca_ports* are routed through independent spines
1: **for** *swleaf = 0* **to** *leaf_sw_num* **do**
2: **for** *swleaf.port = 0* **to** *max_ports* **do**
3: *hca_node = swleaf.port -> remote_node*
4: **if** *hca_node.routed == true* **then**
5: *continue*
6: **end if**
7: *route hcas(hca_node)*
8: **end for**
9: **end for**

[0040] As shown in the above, Algorithm 2, which is a multi-homed routing algorithm, can iterate over all leaf switches, and then can iterate over all leaf switch ports for each leaf switch (Lines 1-9). Thus, Algorithm 2 can be deterministic, which is similar to Algorithm 1.

[0041] Furthermore, Algorithm 2 can take a switch port on a leaf switch in order to find an end node that is associated with the switch port (Line 3). Unlike Algorithm 1, which simply takes the LID of the remote port connected to the leaf switch, Algorithm 2 can take the end node as a parameter for performing the routing calculation (Line 7).

[0042] The following Algorithm 3 is an exemplary algorithm for routing a single end-node in a fat-tree.

Algorithm 3 *route_hcas(hca) function*

Require: Node that is to be routed

Ensure: All *hca_ports* belonging to the node with *hca_lid* are routed
1: **for** *hca_node.port = 0* **to** *port_num* **do**
2: *hca_lid = hca_node.port -> lid*
3: *swleaf = hca_node.port -> remote_node*
4: *swleaf.port = hca_node.port -> remote_port_number*
5: *swleaf.routing_table[hca_lid] = swleaf.port*
6: *route_downgoing_by_going_up()*
7: **end for**
8: hca_node.routed = true
9: clear_redundant_flag()

[0043] As shown in the above, Algorithm 3 can iterate over all ports on a selected end-node (Lines 1-7). For example, Algorithm 3 can route each port on the selected end-node using a modified version of the route_downgoing_by_going_up() function (Line 6). When all ports on the selected end node are routed, the routing algorithm can mark the selected end-node as routed (Line 8), so that end-node is not routed when it is encountered on another leaf switch. Also, Algorithm 3 can improve the performance of the system in various situations (For example, Algorithm 3 can save half of the loop iterations for a two-port node).

[0044] Additionally, Algorithm 3 may be applied to both a scenario with multiple ports on a single host channel adapter (HCA) and a scenario with multiple ports on two or more HCAs. The algorithm can use different methods for identifying ports on the single HCA or on multiple HCAs (or any end ports) on the same logical node. Here, a node can be a physical or virtual server, an IO device, or any kind of end node connected to the IB fabric via one or more HCA ports.

[0045] Furthermore, Algorithm 3 can route each port on the selected node and mark each switch on the path using a flag. Thus, Algorithm 3 can choose different switches for different ports on the same end node. Afterwards, the algorithm can flip the flag on all switches so that the algorithm can progress on the next node.

[0046] Additionally, the system can be optimized by clearing the switch redundancy flag in the clear_redundant_flag() function (Line 9). Instead of using a loop in this function, which iterates over all the switches regardless whether a particular switch was on the path or not, an optimized way for clearing the switch redundancy flag is to create a list of switches that are on the path and making sure that the clear_redundant_flag() function only iterates on those switches in the list.

[0047] The following Algorithm 4 is an exemplary algorithm for routing a single end-node port in a fat-tree.

Algorithm 4 *route_downgoing_by_going_up() function*
Require: Current hop switch
Ensure: Best effort is done to find an upward redundant switch
Ensure: Switches on the path are marked with a redundant
1: *groupmin = 0*
2: *redundant_group = 0*
3: **for** *port_group = 0* **to** *port_group_num* **do**

```
4:      if groupmin == 0 then
5:          if groupmin -> remote_node.redundant then
6:              groupmin = port_group
7:          end if
8:      else if port_group.cntdown < groupmin.cntdown then
9:          groupmin = port_group
10:         if groupmin -> remote_node.redundant then
11:             min redundant group = groupmin
12:         end if
13:     end if
14: end for
15: if groupmin == 0 then
16:     fallback_normal_routing(hca_lid)
17: else if groupmin -> remote node.redundant then
18:     groupmin = min_redundant_group
19:     groupmin -> remote_node.redundant = false
20: end if
```

[0048] As shown in the above, the modified version of the route_downgoing_by_going_up() function in Algorithm 4 treats redundancy as the primary consideration. Unlike Algorithm 1, in which the port group with the lowest downward counters is selected, Algorithm 4 may only choose an upward node of an end node as the next-hop if it does not route any other ports belonging to the end-node (i.e., when the redundant flag is true).

[0049] Here, the redundant flag is cleared before the next end-node is routed. If there are no nodes that are redundant as may happen in heavily oversubscribed fabrics or in case of link failures, mFtree falls back to normal fat-tree routing, in which case a user may observe that this routing function performs similarly to the routing function presented in Algorithm 1.

[0050] Additionally, when there are no alternative switches and parallel links exist between two switches, the above Algorithm 4 is able to select different links for different target ports on the same end node, in order to support for both performance/load-spreading and link level redundancy. On the other hand, in the case when the performance has a priority, the separate links may be chosen for different target ports on the same end node only when both links have the same level of load.

[0051] **Figure 3** shows an illustration of providing redundancy for supporting a fat-tree routing in a network environment. As shown in Figure 3, one or more multi-homed end nodes 301-304 can be connected to a network fabric 300. The network fabric 300, which is based on a fat-tree topology, can include a plurality of leaf switches 311-312, and multiple spine switches or root switches S1 331 – S4 334. Additionally, the network fabric 300 can include one or more intermediate switches, such as switches 321-324.

[0052] Also as shown in Figure 3, the node 301 can include the ports H1 and H2, the node 302 can include the ports H3 and H4, the node 303 can include the ports H5 and H6, and the node 304 can include the ports H7 and H8. Additionally, the root switch S1 331 can have the

switch ports 1-2, the root switch S2 332 can have the switch ports 3-4, the root switch S3 333 can have the switch ports 5-6, and the root switch S4 334 can have the switch ports 7-8.

[0053] The multi-homed routing, such as the mFtree algorithm, can route each multi-homed node 301-304 in a way that the paths to each port on a node are exclusive, i.e. the mFtree algorithm makes sure that each port on a multi-homed node is reachable through an independent path. Additionally, the mFtree algorithm can improve the network performance.

[0054] Furthermore, in the case of a single multi-homed end-node, the mFtree algorithm can ensure that no single link is shared by paths to any pair of ports belonging to the same end node. Also, when there is concurrent traffic from different source ports to different ports on the same destination node in the network fabric 300, the mFtree algorithm can ensure that the concurrent traffic is not sharing any intermediate link when an alternative route exists.

[0055] Thus, using the mFtree algorithm, a failure of a single device, such as the spine switch S1 331 in the fabric 300, may not cause the node 301 to be disconnected, because the paths to the different ports do not converge at the single spine switch S1 331.

[0056] Additionally, the mFtree algorithm treats each port on a same node as a separate and independent entity. Thus, the mFtree algorithm can route on a node-basis instead of on a port-basis, and the mFtree algorithm can address the different characteristics that different end nodes may have.

[0057] **Figure 4** illustrates an exemplary flow chart for supporting multi-homed routing in a network environment, in accordance with an embodiment of the invention. As shown in Figure 4, at step 401, the system can provide an end node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports. Then, at step 402, the system can perform routing for each said port on the end node. Furthermore, at step 403, the system can ensure that the plurality of ports on the end node take mutually independent paths.

[0058] **Figure 5** illustrates a block diagram of a computer system upon which an embodiment of the invention may be implemented. Computer system 500 includes a bus or other communication mechanism for communicating information, and a hardware processor coupled with bus for processing information. Hardware processor may be, for example, a general purpose microprocessor.

[0059] Computer system 500 also includes a main memory, such as a random access memory (RAM) or other dynamic storage device, coupled to bus for storing information and instructions to be executed by the processor. Main memory also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by the processor. Such instructions, when stored in non-transitory storage media accessible to processor, render computer system 500 into a special-purpose machine that is customized to perform the operations specified in the instructions.

[0060] Computer system 500 further includes a read only memory (ROM) or other static storage device coupled to bus for storing static information and instructions for processor. A storage device, such as a magnetic disk or optical disk, is provided and coupled to bus for storing information and instructions.

[0061] Computer system 500 may comprise or be coupled via bus to a display, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device, including alphanumeric and other keys or cursor control, is coupled to bus for communicating information and command selections to processor.

[0062] Computer system 500 may implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs computer system 500 to be a special-purpose machine. According to one embodiment, the techniques herein are performed by computer system 500 in response to the processor executing one or more sequences of one or more instructions contained in main memory. Such instructions may be read into main memory from another storage medium, such as storage device. Execution of the sequences of instructions contained in main memory causes processor to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions.

[0063] Computer system 500 also includes a communication interface coupled to bus. Communication interface may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0064] As shown in **Figure 5**, the storage device may contain a subnet manager, which may be configured to associate an end node with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports. The subnet manager may be further configured to perform routing for each said port on the end node, and ensure the plurality of ports on the end node taking mutually independent paths.

[0065] In some embodiments, the subnet manager may be further configured to mark each switch on a path associated with a port in the plurality of ports on the end node and prevent another path associated with another port in the plurality of ports on the end node from using the marked switches.

[0066] In some embodiments, the subnet manager may be further configured to allow another path associated with another port in the plurality of ports on the end node to use one or

more marked switches when there is no redundant switch, and select independent links for different target ports on the same end node, when parallel links exist on the one or more marked switches.

[0067] In some embodiments, the subnet manager may be further configured to unmark each marked switch after completing the routing of the plurality of ports on the end node.

[0068] In some embodiments, the subnet manager may be further configured to mark the end node as a routed end node after completing the routing of the plurality of ports on the end node.

[0069] In some embodiments, the subnet manager may be further configured to prevent the end node from being routed again, when the end node encounters another leaf switch.

[0070] **Figure 6** illustrates block diagram of a system 600 for implementing an embodiment of the invention. The blocks of the system 600 may be implemented by hardware, software, or a combination of hardware and software to carry out the principles of the invention. It is understood by persons of skill in the art that the blocks described in **Figure 6** may be combined or separated into sub-blocks to implement the principles of the invention as described above. Therefore, the description herein may support any possible combination or separation or further definition of the functional blocks described herein.

[0071] As shown in **Figure 6**, the system 600 may comprise a providing unit 601 configured to provide an end node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports. The system 600 may further comprise a perfoming unit 602 configured to perform routing for each said port on the end node. Furthermore, The system 600 may further comprise a ensuring unit 603 configured to ensure that the plurality of ports on the end node take mutually independent paths.

[0072] Although not shown in **Figure 6**, in some embodiments, the system 600 may further comprise a first allowing unit configured to allow the network fabric to be based on a fat-tree topology.

[0073] In some embodiments, the system 600 may further comprise a second allowing unit configured to allow the end node to be a multi-homed node that is connected to two or more parts of the fat-tree topology through multiple ports.

[0074] In some embodiments, the system 600 may further comprise a first marking unit configured to mark each switch on a path associated with a port in the plurality of ports on the end node, and a first preventing unit configured to prevent another path associated with another port in the plurality of ports on the end node from using the marked switches.

[0075] In some embodiments, the system 600 may further comprise a third allowing unit configured to allow another path associated with another port in the plurality of ports on the end node to use one or more marked switches when there is no redundant switch, and a selecting unit configured to select independent links for different target ports on the same end node, when

parallel links exist on the one or more marked switches.

[0076] In some embodiments, the system 600 may further comprise an unmarking unit configured to unmark each marked switch after completing the routing of the plurality of ports on the end node.

[0077] In some embodiments, the system 600 may further comprise an associating unit configured to associate each mutually independent path from the end node with a different spine switch.

[0078] In some embodiments, the system 600 may further comprise a second marking unit configured to mark the end node as a routed end node after completing the routing of the plurality of ports on the end node.

[0079] In some embodiments, the system 600 may further comprise a second preventing unit configured to prevent the end node from being routed again, when the end node encounters another leaf switch.

[0080] In some embodiments, the system 600 may further comprise a fourth allowing unit configured to allow the routing algorithm to take the end node as a parameter.

[0081] A means for supporting multi-homed routing in a network environment, comprising providing an end node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports; performing routing for each said port on the end node; and ensuring the plurality of ports on the end node take mutually independent paths for said routing.

[0082] A means for supporting multi-homed routing in a network environment wherein the network fabric is based on a fat-tree topology.

[0083] A means for supporting multi-homed routing in a network environment wherein the end node is a multi-homed node that is connected to two or more parts of the fat-tree topology through multiple ports.

[0084] A means for supporting multi-homed routing in a network environment further comprising marking each switch on a path associated with a port in the plurality of ports on the end node; and preventing another path associated with another port in the plurality of ports on the end node from using the marked switches.

[0085] A means for supporting multi-homed routing in a network environment further comprising allowing another path associated with another port in the plurality of ports on the end node to use one or more marked switches when there is no redundant switch, and selecting independent links for different target ports on the same end node, when parallel links exist on the one or more marked switches.

[0086] A means for supporting multi-homed routing in a network environment further comprising unmarking each marked switch after completing the routing of the plurality of ports on the end node.

[0087] A means for supporting multi-homed routing in a network environment further comprising associating each mutually independent path from the end node with a different spine switch.

[0088] A means for supporting multi-homed routing in a network environment further comprising marking the end node as a routed end node after completing the routing of the plurality of ports on the end node.

[0089] A means for supporting multi-homed routing in a network environment further comprising preventing the end node from being routed again, when the end node encounters another leaf switch.

[0090] A means for supporting multi-homed routing in a network environment further comprising using a routing algorithm which takes the end node as a parameter.

[0091] The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[0092] In some embodiments, the present invention includes a computer program product which is a storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0093] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The modification and variation include any relevant combination of the described features. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

Claims:

What is claimed is:

1. A method for supporting multi-homed routing in a network environment, comprising:

providing an end node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports;

performing routing for each said port on the end node; and

ensuring the plurality of ports on the end node take mutually independent paths for said routing.

2. The method according to Claim 1, further comprising:

wherein the network fabric is based on a fat-tree topology.

3. The method according to Claim 2, further comprising:

wherein the end node is a multi-homed node that is connected to two or more parts of the fat-tree topology through multiple ports.

4. The method according to any preceding Claim, further comprising:

marking each switch on a path associated with a port in the plurality of ports on the end node; and

preventing another path associated with another port in the plurality of ports on the end node from using the marked switches.

5. The method according to Claim 4, further comprising:

allowing another path associated with another port in the plurality of ports on the end node to use one or more marked switches when there is no redundant switch, and

selecting independent links for different target ports on the same end node, when parallel links exist on the one or more marked switches.

6. The method according to Claim 4 or 5, further comprising:

unmarking each marked switch after completing the routing of the plurality of ports on the end node.

7. The method according to any preceding Claim, further comprising:

associating each mutually independent path from the end node with a different spine switch.

8. The method according to any preceding Claim, further comprising:

marking the end node as a routed end node after completing the routing of the plurality of ports on the end node.

9. The method according to Claim 8, further comprising:

preventing the end node from being routed again, when the end node encounters another leaf switch.

10. The method according to any preceding Claim, further comprising:

using a routing algorithm which takes the end node as a parameter.

11. A computer program comprising instructions that when executed by a system cause the system to perform the method of any preceding claim.

12. A non-transitory machine readable storage medium having the computer program of claim 11 sotred thereon.

13. A non-transitory machine readable storage medium having instructions stored thereon that when executed cause a system to perform the steps comprising:

providing an edn node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports;

performing routing for each said port on the end node; and

ensuring the plurality of ports on the end node taking mutually independent paths.

14. A system for supporting multi-homed routing in a network environment, comprising:

one or more microprocessors,

a subnet manager running on the one or more microprocessors, wherein the subnet manager operates to

associate an end node with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports;

perform routing for each said port on the end node; and

ensure the plurality of ports on the end node take mutually independent paths for said routing.

15. The system according to Claim 14, wherein:

the network fabric is based on a fat-tree topology.

16. The system according to any of Claims 15, wherein:

the end node is a multi-homed node that is connected to two or more parts of the fat-tree topology through multiple ports.

17. The system according to any of Claims 14 to 16, wherein:

the subnet manager operates to:

mark each switch on a path associated with a port in the plurality of ports on the end node; and

prevent another path associated with another port in the plurality of ports on the end node from using the marked switches.

18. The system according to Claim 14, wherein:

the subnet manager operates to:

allow another path associated with another port in the plurality of ports on the end node to use one or more marked switches when there is no redundant switch, and

select independent links for different target ports on the same end node, when parallel links exist on the one or more marked switches.

19. The system according to Claim 17 or 18, wherein:

the subnet manager operates to unmark each marked switch after completing the routing of the plurality of ports on the end node.

20. The system according to any of Claims 14 to 19, wherein:

each mutually independent path from the end node is associated with a different spine switch.

21. The system according to any of Claims 14 to 20, wherein:

the subnet manager operates to mark the end node as a routed end node after completing the routing of the plurality of ports on the end node.

22. The system according to Claim 21, wherein:

the subnet manager operates to prevent the end node from being routed again, when the end node encounters another leaf switch.

23. The system accoridng to any of Claim 14 to 22, wherein:

the subnet manager operates to use a routing algorithm which takes the end node as a parameter.

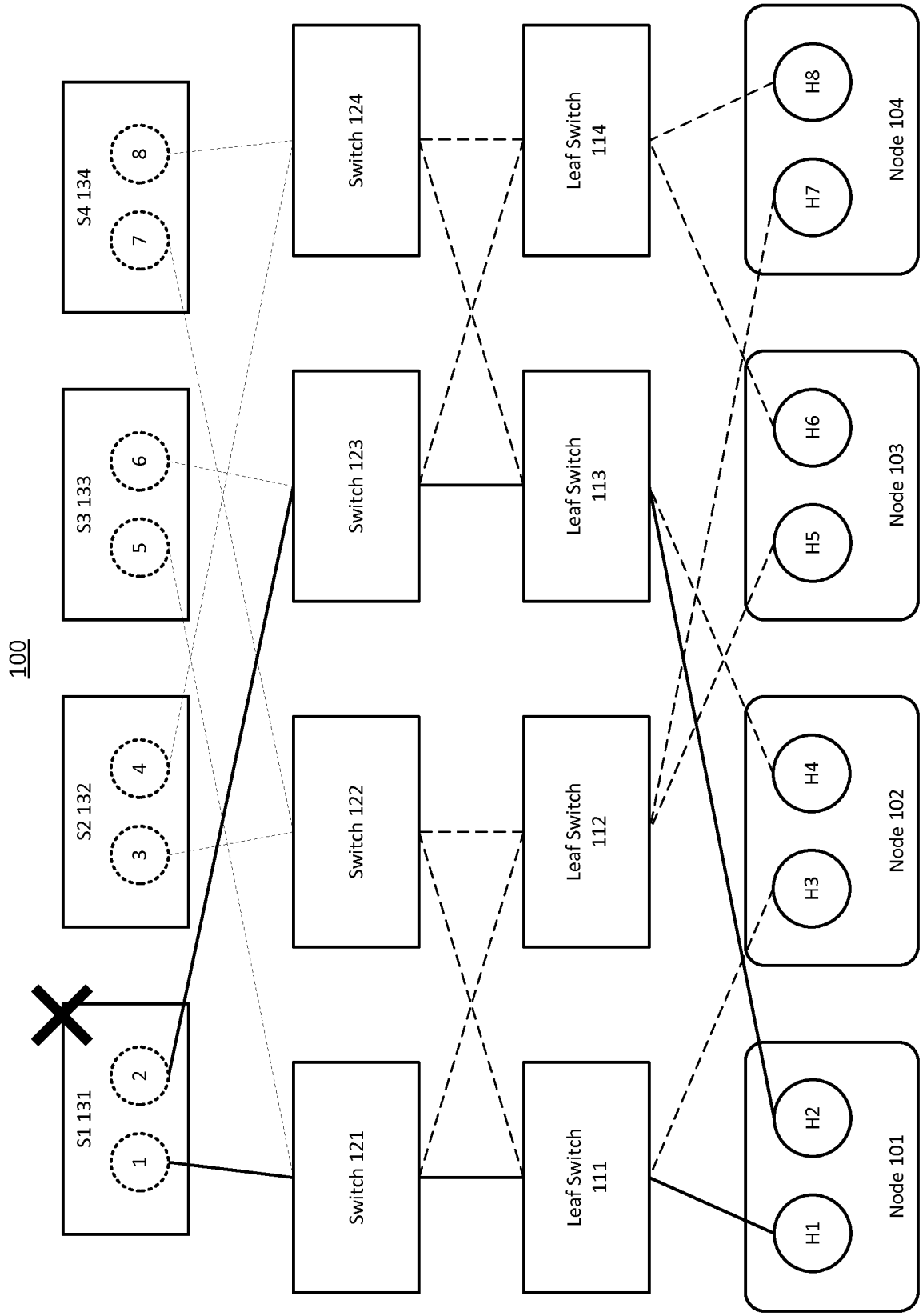
100
S1 131
1
2
S2 132
3
4
S3 133
5
6
S4 134
7
8
Switch 121
Switch 122
Switch 123
Switch 124
Leaf Switch 111
Leaf Switch 112
Leaf Switch 113
Leaf Switch 114
Node 101
H1
H2
Node 102
H3
H4
Node 103
H5
H6
Node 104
H7
H8

FIGURE 1

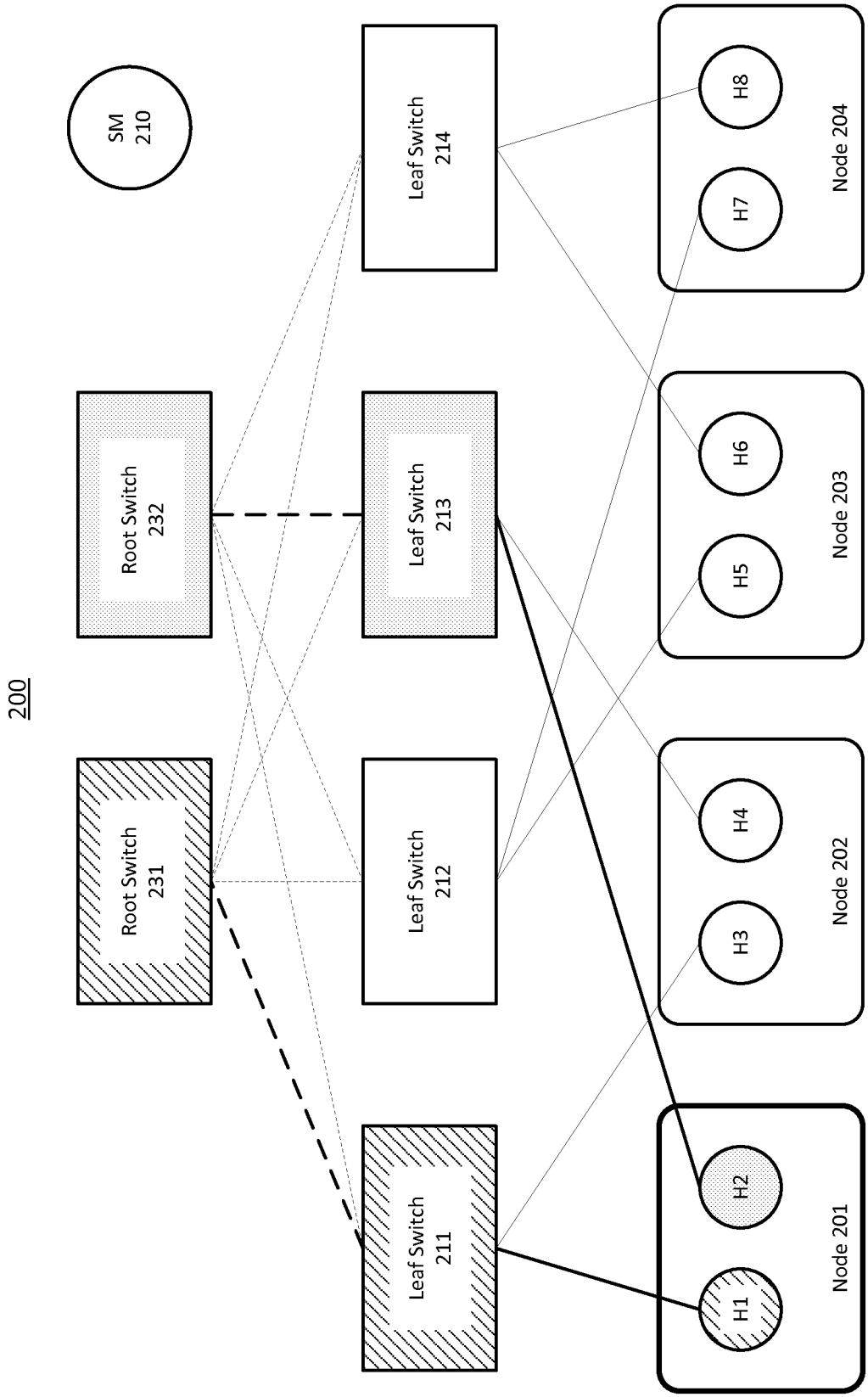
200
SM 210
Root Switch 232
Root Switch 231
Leaf Switch 214
Leaf Switch 213
Leaf Switch 212
Leaf Switch 211
Node 204
H8
H7
Node 203
H6
H5
Node 202
H4
H3
Node 201
H2
H1

FIGURE 2

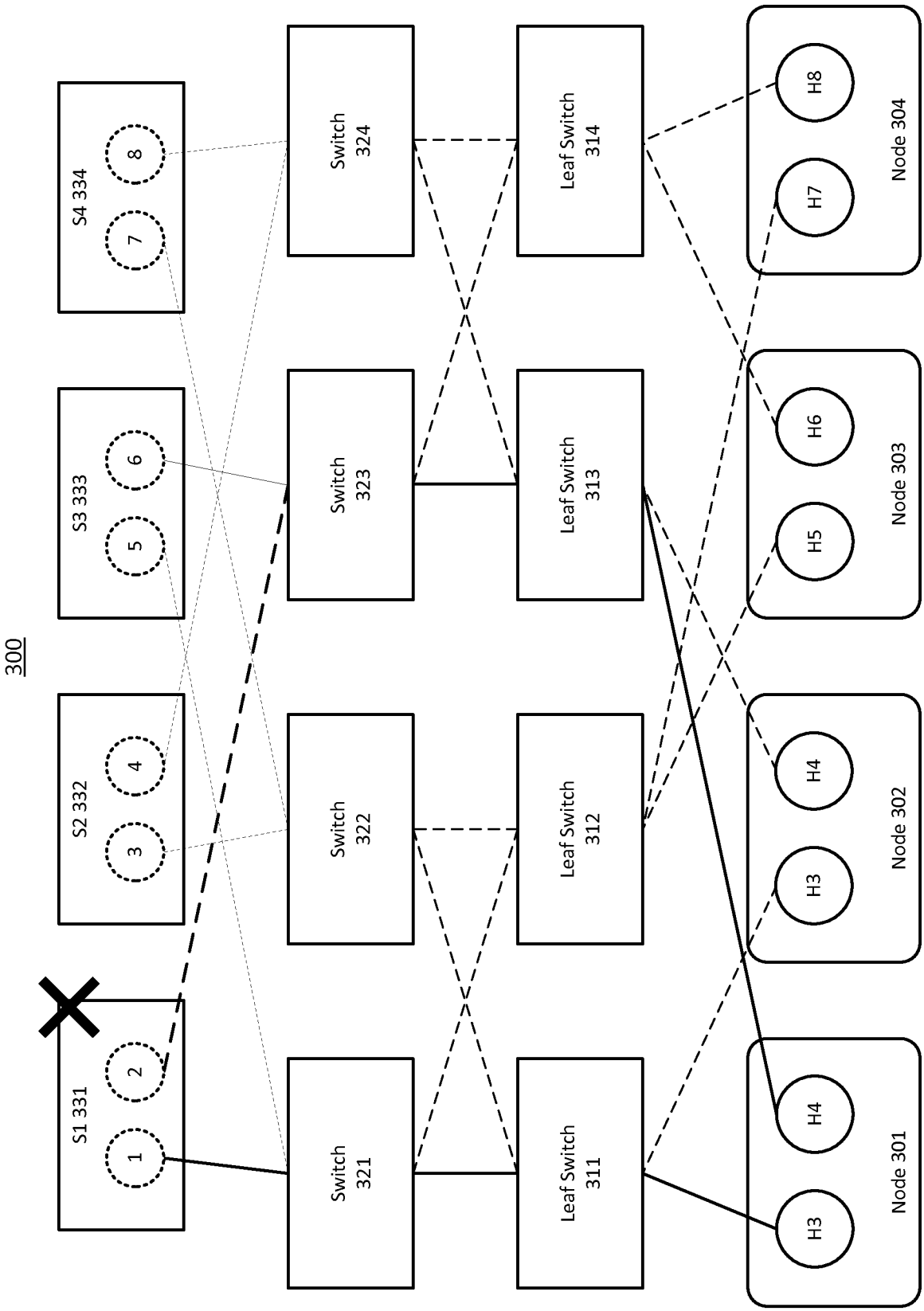
300
S1 331
1
2
S2 332
3
4
S3 333
5
6
S4 334
7
8
Switch 321
Switch 322
Switch 323
Switch 324
Leaf Switch 311
Leaf Switch 312
Leaf Switch 313
Leaf Switch 314
Node 301
H3
H4
Node 302
H3
H4
Node 303
H5
H6
Node 304
H7
H8

FIGURE 3

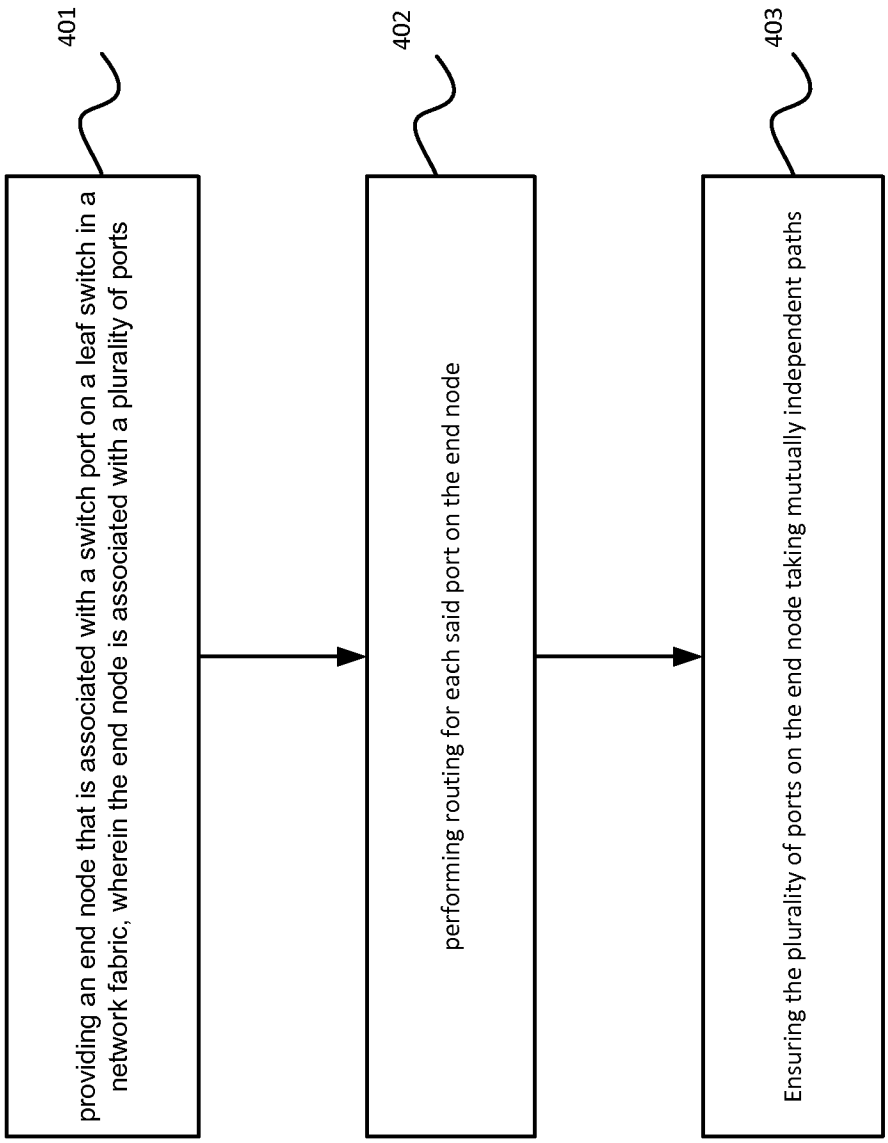
401
providing an end node that is associated with a switch port on a leaf switch in a network fabric, wherein the end node is associated with a plurality of ports
402
performing routing for each said port on the end node
403
Ensuring the plurality of ports on the end node taking mutually independent paths

FIGURE 4

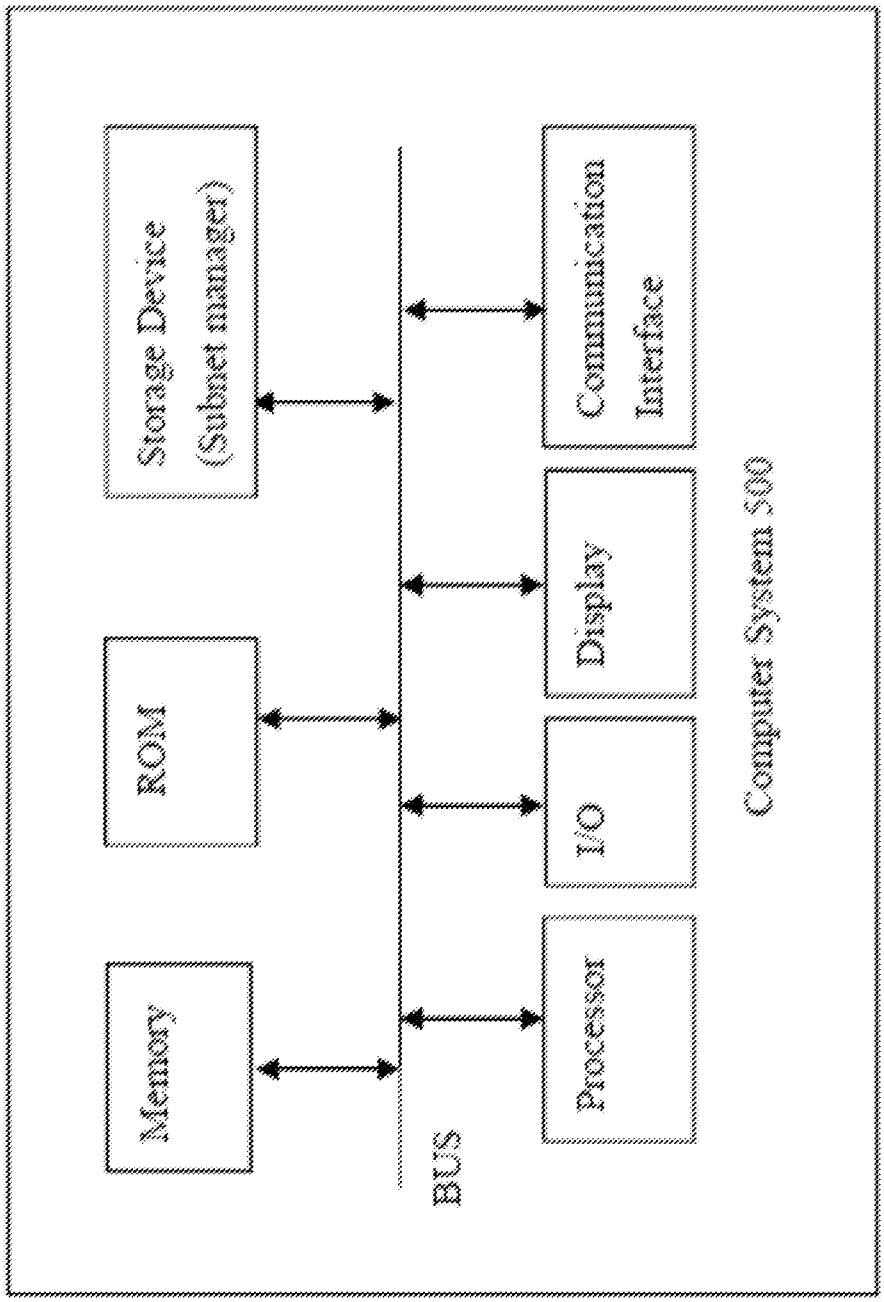
Computer System 500
Storage Device
(Subnet manager)
ROM
Memory
BUS
Processor
I/O
Display
Communication
Interface

FIGURE 5

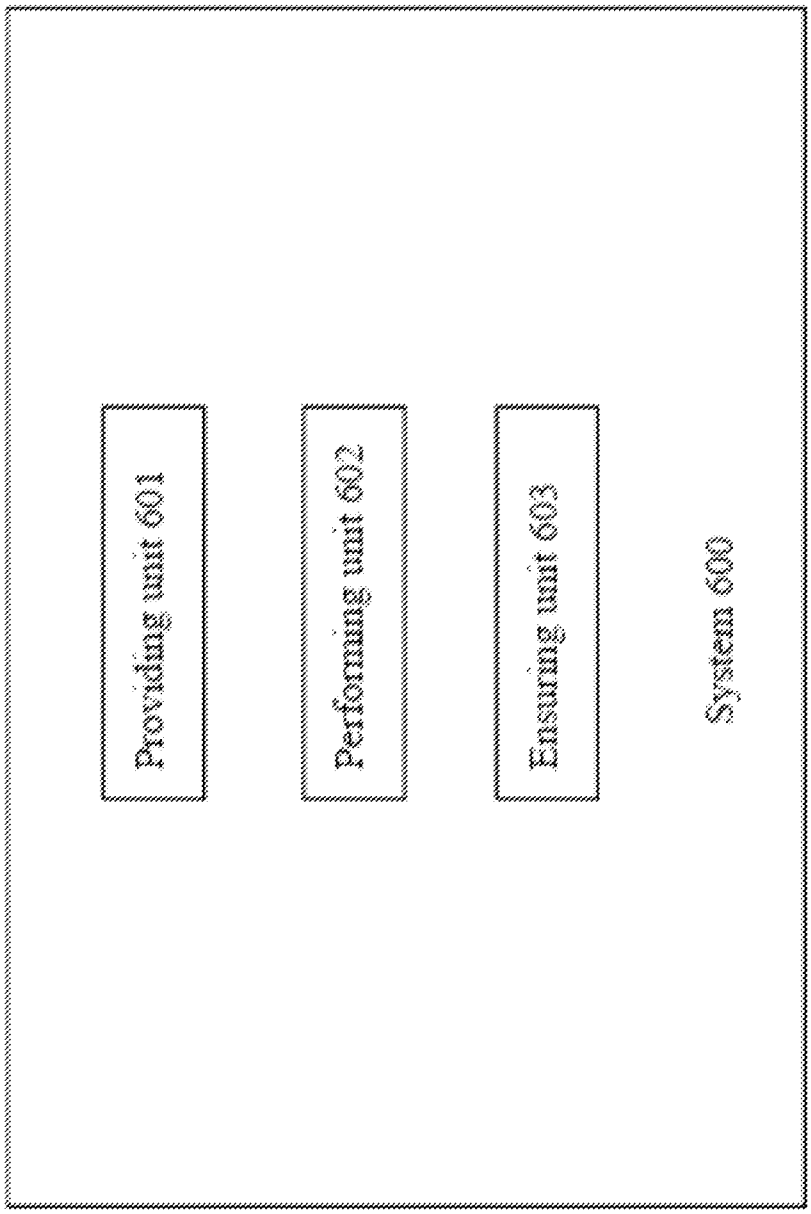
System 600
Providing unit 601
Performing unit 602
Ensuring unit 603

FIGURE 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/047280

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L12/735 H04L12/931 H04L12/707 H04L12/933
ADD.

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

B. FIELDS SEARCHED

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

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 2012/027017 A1 (RAI SMITA [US] ET AL) 2 February 2012 (2012-02-02) paragraphs [0002], [0003], [0019] - [0021], [0023] - [0035] -----	1,11-14 2-10, 15-23
Y	US 2013/114620 A1 (BOGDANSKI BARTOSZ [NO]) 9 May 2013 (2013-05-09) paragraphs [0045] - [0054] -----	4-10, 17-23
Y	LAIQUAN HAN ET AL: "A Novel Multipath Load Balancing Algorithm in Fat-Tree Data Center", 1 December 2009 (2009-12-01), CLOUD COMPUTING, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 405 - 412, XP019135548, ISBN: 978-3-642-10664-4 sections 3.1 and 3.2 -----	2-10, 15-23
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 November 2014	14/11/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Ramenzoni, Stefano

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/047280

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	REINEMO SVEN-ARNE ET AL: "Multi-homed Fat-Tree Routing with InfiniBand", 2014 22ND EUROMICRO INTERNATIONAL CONFERENCE ON PARALLEL, DISTRIBUTED, AND NETWORK-BASED PROCESSING, IEEE, 12 February 2014 (2014-02-12), pages 122-129, XP032584966, ISSN: 1066-6192, DOI: 10.1109/PDP.2014.22 [retrieved on 2014-04-11] the whole document --- BOGDANSKI B ET AL: "sFtree: A fully connected and deadlock free switch-to-switch routing algorithm for fat-trees", ACM TRANSACTIONS ON ARCHITECTURE AND CODE OPTIMIZATION, ASSOCIATION FOR COMPUTING MACHINERY, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA , vol. 8, no. 4 14 January 2012 (2012-01-14), pages 1-20, XP002692976, ISSN: 1544-3566, DOI: 10.1145/2086696.2086734 Retrieved from the Internet: URL:http://simula.no/publications/Simula.simula.864 [retrieved on 2013-02-27] the whole document ---	1-23
A		1-23

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/047280

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012027017 A1	02-02-2012	NONE	
US 2013114620 A1	09-05-2013	CN 103907322 A	02-07-2014
		EP 2777229 A1	17-09-2014
		US 2013114620 A1	09-05-2013
		WO 2013071130 A1	16-05-2013