

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
29 March 2007 (29.03.2007)

PCT

(10) International Publication Number
WO 2007/035307 A2

(51) International Patent Classification:

H04L 12/28 (2006.01)

(21) International Application Number:

PCT/US2006/035294

(22) International Filing Date:

11 September 2006 (11.09.2006)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

11/228,162 16 September 2005 (16.09.2005) US

(71) Applicant (for all designated States except US): **CISCO TECHNOLOGY, INC.** [US/US]; 170 West Tasman Drive, San Jose, CA 95134 (US).

(72) Inventor: **TALLET, Francois, Edouard**; 2205 Gianera Street, Santa Clara, CA 95054 (US).

(74) Agents: **BARBAS, Charles, J.** et al.; Cesari and McKenna, LLP, 88 Black Falcon Avenue, Boston, MA 02210 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, 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.

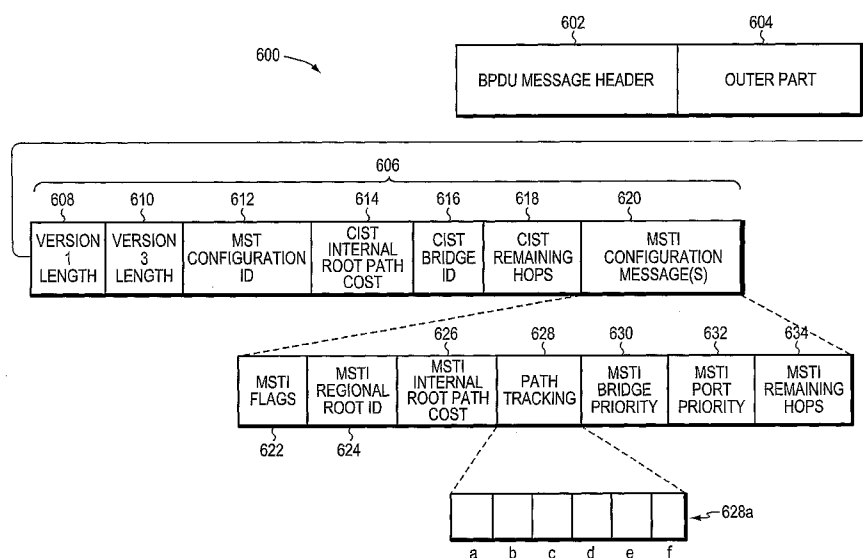
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, 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

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR GENERATING SYMMETRICAL SPANNING TREES



(57) Abstract: A system and method creates multiple, symmetric spanning trees within a network. Bridges within the network generate, send and process Spanning Tree Protocol (STP) control messages that are updated as they are propagated across the network to re-fleet the paths followed by the messages. The bridges, moreover, utilize the path indication value of received STP control messages to compute the spanning trees. The path indication values are preferably derived from the sum of Bridge Identifiers (IDs) corresponding to the bridges through which the STP control message has passed from the root bridge to the current bridge processing the STP control message. Each bridge also tags newly received messages with the Virtual Local Area Network (VLAN) identifier (VID) associated with the spanning tree for which the bridge is the root, thereby causing the messages to follow more optimal paths through the network.

WO 2007/035307 A2

SYSTEM AND METHOD FOR GENERATING SYMMETRICAL SPANNING TREES

BACKGROUND OF THE INVENTION

Field of the Invention

5 The present invention relates generally to computer networks, and more specifically, to a method and apparatus for efficiently transmitting messages across the network.

Background Information

Many organizations, including businesses, governments and educational institutions, utilize computer networks so that employees and others may share and exchange
10 information and/or resources. A computer network typically comprises a plurality of entities interconnected by means of one or more communications media. An entity may consist of any device, such as a computer, that “sources” (i.e., transmits) or “sinks” (i.e., receives) data frames over the communications media. A common type of computer network is a local area network (“LAN”) which typically refers to a privately owned network within a single building or campus. LANs typically employ a data communication
15 protocol (LAN standard), such as Ethernet, or a wireless protocol, that defines the functions performed by data link and physical layers of a communications architecture (i.e., a protocol stack).

One or more intermediate network devices are often used to couple LANs together and allow the corresponding entities to exchange information. For example, a
20 bridge may be used to provide a “switching” function between two or more LANs or end stations. Typically, the bridge is a computer and includes a plurality of ports that are coupled via LANs either to other bridges, or to end stations such as routers or host computers. Ports used to couple bridges to each other are generally referred to as a trunk
25 ports, whereas ports used to couple bridges to end stations are generally referred to as access ports. The bridging function includes receiving data from a sending entity at a

source port and transferring that data to at least one destination port for forwarding to one or more receiving entities.

Spanning Tree Algorithm

Most computer networks include redundant communications paths so that a failure of any given link does not isolate any portion of the network. Such networks are typically referred to as meshed or partially meshed networks. The existence of redundant links, however, may cause the formation of circuitous paths or “loops” within the network. Loops are highly undesirable because data frames may traverse the loops indefinitely.

Furthermore, some devices, such as bridges or switches, replicate frames whose destination is not known resulting in a proliferation of data frames along loops. The resulting traffic can overwhelm the network. Other intermediate devices, such as routers, that operate at higher layers within the protocol stack, such as the Internetwork Layer of the Transmission Control Protocol/Internet Protocol (“TCP/IP”) reference model, deliver data frames and learn the addresses of entities on the network differently than most bridges or switches, such that routers are generally not susceptible to sustained looping problems.

To avoid the formation of loops, most bridges and switches execute a spanning tree protocol which allows them to calculate an active network topology that is loop-free (i.e., a tree) and yet connects every pair of LANs within the network (i.e., the tree is spanning). The IEEE promulgated a standard (IEEE Std. 802.1D-1998™) that defines a spanning tree protocol to be executed by 802.1D compatible devices. In general, by executing the 802.1D spanning tree protocol, bridges elect a single bridge within the bridged network to be the “Root Bridge”. The 802.1D standard takes advantage of the fact that each bridge has a unique numerical identifier (bridge ID) by specifying that the Root Bridge is the bridge with the lowest bridge ID. In addition, for each LAN coupled to any bridge, exactly one port (the “Designated Port”) on one bridge (the “Designated Bridge”) is elected. The Designated Bridge is typically the one closest to the Root Bridge. All ports on the Root Bridge are Designated Ports, and the Root Bridge is the Designated Bridge on all the LANs to which it has ports.

Each non-Root Bridge also selects one port from among its non-Designated Ports (its "Root Port") which gives the lowest cost path to the Root Bridge. The Root Ports and Designated Ports are selected for inclusion in the active topology and are placed in a *forwarding* state so that data frames may be forwarded to and from these ports and thus onto the LANs interconnecting the bridges and end stations of the network. Ports not included within the active topology are placed in a *blocking* state. When a port is in the blocking state, data frames will not be forwarded to or received from the port. A network administrator may also exclude a port from the spanning tree by placing it in a *disabled* state.

To obtain the information necessary to run the spanning tree protocol, bridges exchange special messages called *configuration bridge protocol data unit* (BPDU) messages or simply BPDUs. BPDUs carry information, such as assumed root and lowest root path cost, used in computing the active topology. More specifically, upon start-up, each bridge initially assumes itself to be the Root Bridge and transmits BPDUs accordingly. Upon receipt of a BPDU from a neighboring device, its contents are examined and compared with similar information (e.g., assumed root and lowest root path cost) stored by the receiving bridge in memory. If the information from the received BPDU is "better" than the stored information, the bridge adopts the better information and uses it in the BPDUs that it sends (adding the cost associated with the receiving port to the root path cost) from its ports, other than the port on which the "better" information was received. Although BPDUs are not forwarded by bridges, the identifier of the Root Bridge is eventually propagated to and adopted by all bridges as described above, allowing them to select their Root Port and any Designated Port(s).

In order to adapt the active topology to changes and failures, the Root Bridge periodically (e.g., every *hello time*) transmits BPDUs. In response to receiving BPDUs on their Root Ports, bridges transmit their own BPDUs from their Designated Ports, if any. Thus, BPDUs are periodically propagated throughout the bridged network, confirming the active topology. As BPDU information is updated and/or timed-out and the active topology is re-calculated, ports may transition from the blocking state to the forwarding state and vice versa. That is, as a result of new BPDU information, a previously blocked port may learn that it should be in the forwarding state (e.g., it is now the Root Port or a Designated Port).

Rapid Spanning Tree Protocol

Recently, the IEEE issued a new version of the 802.1D standard, known as IEEE Std. 802.1D-2004, that describes a rapid spanning tree protocol (RSTP) to be executed by otherwise 802.1D compatible devices. The RSTP similarly selects one bridge of a
5 bridged network to be the root bridge and defines an active topology that provides complete connectivity among the LANs while severing any loops. Each individual port of each bridge is assigned a port role according to whether the port is to be part of the active topology. The port roles defined by the 802.1D-2004 standard include Root, Designated, Alternate and Backup. The bridge port offering the best, e.g., lowest cost, path to the root
10 is assigned the Root Port Role. Each bridge port offering an alternative, e.g., higher cost, path to the root is assigned the Alternate Port Role. Each bridge port providing the lowest cost path from a given LAN is assigned the Designated Port Role, while all other ports coupled to the given LAN in loop-back fashion are assigned the Backup Port Role.

Those ports that have been assigned the Root Port and Designated Port Roles are
15 placed in the forwarding state, while ports assigned the Alternate and Backup Roles are placed in a state. A port assigned the Root Port Role can be rapidly transitioned to the forwarding state provided that all of the ports assigned the Alternate Port Role are placed in the blocking state. Similarly, if a failure occurs on the port currently assigned the Root Port Role, a port assigned the Alternate Port Role can be reassigned to the Root Port Role
20 and rapidly transitioned to the forwarding state, provided that the previous Root Port has been transitioned to the discarding or blocking state. A port assigned the Designated Port Role or a Backup Port that is to be reassigned to the Designated Port Role can be rapidly transitioned to the forwarding state, provided that the roles of the ports of the downstream bridge are consistent with this port being assigned the Designated Port Role. The RSTP
25 provides an explicit handshake to be used by neighboring bridges to confirm that a new Designated Port can rapidly transition to the forwarding state.

Like the STP described in the 1998 version of the 802.1D specification standard, bridges running RSTP also exchange BPDUs in order to determine which roles to assign to the bridge's ports. The BPDUs are also utilized in the handshake employed to rapidly
30 transition Designated Ports to the forwarding state.

Virtual Local Area Networks

A computer network may also be segmented into a series of logical networks. For example, U.S. Patent No. 5,394,402, issued February 28, 1995 to Ross (the “’402 Patent”), discloses an arrangement for associating any port of a switch with any particular network segment. Specifically, according to the ’402 Patent, any number of physical ports of a particular switch may be associated with any number of groups within the switch by using a virtual local area network (VLAN) arrangement that *virtually* associates the port with a particular VLAN designation. More specifically, the switch or hub associates VLAN designations with its ports and further associates those VLAN designations with messages transmitted from any of the ports to which the VLAN designation has been assigned.

The VLAN designation for each port is stored in a memory portion of the switch such that every time a message is received on a given access port the VLAN designation for that port is associated with the message. Association is accomplished by a flow processing element which looks up the VLAN designation in the memory portion based on the particular access port at which the message was received. In many cases, it may be desirable to interconnect a plurality of these switches in order to extend the VLAN associations of ports in the network. Those entities having the same VLAN designation function as if they are all part of the same LAN. VLAN-configured bridges are specifically configured to prevent message exchanges between parts of the network having different VLAN designations in order to preserve the boundaries of each VLAN. Nonetheless, intermediate network devices operating above L2, such as routers, can relay messages between different VLAN segments.

In addition to the ’402 Patent, the IEEE promulgated the 802.1Q specification standard for Virtual Bridged Local Area Networks. To preserve VLAN associations of messages transported across trunks or links in VLAN-aware networks, both Ross and the IEEE Std. 802.1Q-2003 specification standard disclose appending a VLAN identifier (VID) field to the corresponding frames. In addition, U.S. Patent No. 5,742,604 to Edsall et al. (the “’604 patent”), which is commonly owned with the present application, discloses an Interswitch Link (ISL) encapsulation mechanism for efficiently transporting

packets or frames, including VLAN-modified frames, between switches while maintaining the VLAN association of the frames. In particular, an ISL link, which may utilize the Fast Ethernet standard, connects ISL interface circuitry disposed at each switch. The transmitting ISL circuitry encapsulates the frame being transported within an ISL header and ISL error detection information, while the ISL receiving circuitry strips off this information and recovers the original frame.

Multiple Spanning Tree Protocol

Within the IEEE Std. 802.1Q-2003, the IEEE also included a specification standard for a Spanning Tree Protocol that is specifically designed for use with networks that support VLANs. The Multiple Spanning Tree Protocol (MSTP), which is described in the IEEE 802.1s supplement to IEEE Std. 802.1Q-2003, organizes a bridged network into regions. Within each region, MSTP establishes an Internal Spanning Tree (IST) which provides connectivity to all bridges within the respective region and to the ISTs established within other regions. The IST established within each MSTP Region also provides connectivity to the one Common Spanning Tree (CST) established outside of the MSTP regions by IEEE Std. 802.1Q-1998 compatible bridges running STP or RSTP. The IST of a given MST Region receives and sends BPDUs to the CST. Accordingly, all bridges of the bridged network are connected by a single Common and Internal Spanning Tree (CIST). From the point of view of the legacy or IEEE Std. 802.1Q-1998 bridges, moreover, each MST Region appears as a single virtual bridge on the CST.

Within each MST Region, the MSTP compatible bridges establish a plurality of active topologies, each of which is called a Multiple Spanning Tree Instance (MSTI). The MSTP bridges also assign or map each VLAN to one and only one of the MSTIs. Because VLANs may be assigned to different MSTIs, frames associated with different VLANs can take different paths through an MSTP Region. The bridges may, but typically do not, compute a separate topology for every single VLAN, thereby conserving processor and memory resources. Each MSTI is basically a simple RSTP instance that exists only inside the respective Region, and the MSTIs do not interact outside of the Region.

MSTP, like the other spanning tree protocols, uses BPDUs to establish the ISTs and MSTIs as well as to define the boundaries of the different MSTP Regions. The bridges do not send separate BPDUs for each MSTI. Instead, every MSTP BPDU carries the information needed to compute the active topology for all of the MSTIs defined within the respective Region. Each MSTI, moreover, has a corresponding Identifier (ID) and the MSTI IDs are encoded into the bridge IDs. That is, each bridge has a unique ID, as described above, and this ID is made up of a fixed portion and a settable portion. With MSTP, the settable portion of a bridge's ID is further organized to include both a settable priority component and a system ID extension. The system ID extension corresponds to the CIST or one of the MSTI IDs. The MSTP compatible bridges within a given Region will thus have a different bridge ID for the CIST and each MSTI. For a given MSTI, the bridge having the lowest bridge ID for that instance is elected the root. Thus, an MSTP compatible bridge may be the root for one MSTI but not another within a given MSTP Region.

Each bridge running MSTP also has a single MST Configuration Identifier (ID) that consists of three attributes: an alphanumeric configuration name, a revision level and a VLAN mapping table that associates each of the potential 4096 VLANs to a corresponding MSTI. Each bridge, moreover loads its MST Configuration ID into the BPDUs sourced by the bridge. Because bridges only need to know whether or not they are in the same MST Region, they do not propagate the actual VLAN to MSTI tables in their BPDUs. Instead, the MST BPDUs carry only a digest of the VLAN to MSTI table or mappings. The digest is generated by applying the well-know MD-5 algorithm to the VLAN to MSTI table. When a bridge receives an MST BPDU, it extracts the MST Configuration ID contained therein, including the digest, and compares it to its own MST Configuration ID to determine whether it is in the same MST Region as the bridge that sent the MST BPDU. If the two MST Configuration IDs are the same, then the two bridges are in the same MST Region. If, however, the two MST Configuration IDs have at least one non-matching attribute, i.e., either different configuration names, different revision levels and/or different computed digests, then the bridge that received the BPDU concludes that it is in a different MST Region than the bridge that sourced the BPDU. A port of an MST bridge, moreover, is considered to be at the boundary of an MST Region

if the Designated Bridge is in a different MST Region or if the port receives legacy BPDUs.

Fig. 1 is a highly schematic block diagram of an MST BPDUs 100 in accordance with IEEE Std. 802.1Q-2003. The MST BPDUs 100 includes a header 102 compatible with the Media Access Control (MAC) layer of the respective LAN standard, e.g., Ethernet. The header 102 comprises a destination address (DA) field, a source address (SA) field, a Destination Service Access Point (DSAP) field, and a Source Service Access Point (SSAP), among others. The DA field carries a unique bridge multicast destination address assigned to the spanning tree protocol, and the DSAP and SSAP fields carry standardized identifiers assigned to the spanning tree protocol. Appended to header 102 is a BPDUs message area that includes an "outer" part 104 and an "inner" part 106. The outer part 104 has the same format as an RSTP BPDUs message and is recognized as a valid RSTP BPDUs message by bridges that do not implement MSTP. The "inner" part 106 is utilized by bridges executing MSTP to establish the IST and the MSTIs. The inner part 106 has a set of spanning tree parameters, e.g., spanning tree priority vectors, for the IST, and a set of parameters for each MSTI supported by the bridge sourcing the MSTP BPDUs 100.

Outer part 104, also referred to as the CIST priority vector, has a plurality of fields, including a protocol identifier (ID) field 108, a protocol version ID field 110, a BPDUs type field 112, a CIST flags field 114, a CIST root ID field 116, a CIST external path cost field 118, a CIST regional root ID field 120, a CIST port ID field 122, a message age field 124, a maximum (MAX) age field 126, a hello time field 128, and a forward delay field 130. The CIST root identifier field 116 contains the identifier of the bridge assumed to be the root of the Common and Internal Spanning Tree, which may be in the same MSTP Region as the bridge sourcing the BPDUs message 100, in another MSTP Region or in part of the bridged network that is not running MSTP. The CIST external path cost field 118 contains a value representing the lowest cost from the bridge sourcing the BPDUs 100 to the CIST root identified in field 116 without passing through any other bridge in the same region as the bridge that is sourcing the BPDUs message 100.

Inner part 106, also referred to as an MSTI priority vector, similarly has a plurality of fields, including a version 1 length field 132, which is typically set to null, a version 3 length field 136, an MST configuration ID field 138, a CIST internal root path cost field 140, a CIST bridge ID field 144, and a CIST remaining hops field 150. Inner part
5 106 may further include one or more optional MSTI configuration messages 152, each of which constitutes another MSTI priority vector or M-record.

Because version 2 of the RSTP does not specify any additional fields beyond those already specified by version 1, the MST BPDU does not have a version 2 length field.

10 As mentioned above, the MST configuration ID field 138 is made up of three sub-fields: a configuration name sub-field 154, a revision level sub-field 156 and an MD-5 checksum sub-field 158. The configuration name sub-field 154 carries a variable length text string encoded within a fixed size, e.g., 32-octets. The revision level sub-field 156 carries an integer encoded within a fixed field of two octets. The MD-5 checksum sub-
15 field 158 carries a 16-octet signature created by applying the MD-5 algorithm to the bridge's VLAN to MSTI table, which contains 4096 consecutive two octet elements.

Each MSTI Configuration Message 152 consists of a plurality of fields including a MSTI flags field 160, a MSTI regional root ID field 162, a MSTI internal root path cost field 164, a MSTI bridge priority field 166, a MSTI port priority field 168, and a MSTI
20 remaining hops field 170. MST bridges utilize the STP parameters contained in fields 140-150 of inner part 106 and in each MSTI configuration message 152 to compute an active topology for each MSTI configured in the respective region.

Although the STP prevents the formation of loops within a bridged network, it can result in the creation of less than ideal paths. Fig. 2 is a highly schematic illustration
25 of a bridged network 200. Network 200 includes six bridges 202a-f, each having a plurality of ports (not shown). The bridges 202a-f are interconnected by network segments 204, such as shared media or point-to-point links. Suppose that bridge 202a is elected the root of the network 200. Execution of the STP will cause bridge 202b to block one of its ports to prevent the formation of a loop in network 200. Suppose bridge 202b determines
30 that its port coupled to bridge 202f is to be blocked as indicated by the "X" on Fig. 2.

Execution of the STP thus prevented the formation of a loop in network 200. The path utilized by data messages going from bridge 202b to bridge 202f, however, is far from optimal. In particular, data messages received at bridge 202b, e.g., from a Local Area Network (LAN) connected thereto (not shown) that are to be delivered to bridge 202f are
5 routed through bridges 202e, 202d, 202a and 202c. Given that bridges 202b and 202f are neighbors, this roundabout path is obviously not ideal. Indeed, such a situation results in delays and wasted bandwidth.

SUMMARY OF THE INVENTION

Briefly, the invention relates to a system and method for creating multiple, sym-
10 metric spanning trees within a bridged network. The bridges cooperate to establish a plurality of spanning tree instances through the network. In the illustrative embodiment, the bridges form the same number of spanning trees as there are bridges, such that each bridge is the root of one spanning tree instance. The bridges further cooperate to ensure that the paths defined by each spanning tree instance are symmetrical between every pair
15 of bridges in the network. When a network message is received by the first bridge in the network, that bridge forwards the message on the spanning tree instance for which it is the root. In particular, the bridge may tag the message with a Virtual Local Area Network (VLAN) identifier (VID) that is associated with that spanning tree instance and forward the message into the network. The tagged message thus follows a more optimal
20 path from the bridge at which it was initially received to its destination.

To establish the symmetrical spanning trees, the bridges generate, send and process novel Spanning Tree Protocol (STP) control messages. In accordance with the invention, the STP control message includes a new field, which is updated as the message is propagated across the network to reflect the path followed by the message. The bridges,
25 moreover, utilize the value in this new path indication field as part of the spanning tree priority vectors used to compute the active topologies. Specifically, the bridges look to the value of the path record to select among two or more possible root ports, each of which has the same cost to the root. In an illustrative embodiment, the path record carries a value derived from the sum of Bridge Identifiers (IDs) corresponding to the bridges
30 through which the STP control message has passed from the root bridge to the current

bridge processing the STP control message. Because the sum of Bridge IDs along a given path between two bridges will be the same in both directions, utilization of the new path record in the selection of root ports ensures that the spanning trees between each pair of bridges will be symmetric.

5

BRIEF DESCRIPTION OF THE DRAWINGS

The invention description below refers to the accompanying drawings, of which:

Fig. 1, previously discussed, is a schematic block diagram of a conventional configuration bridge protocol data unit in accordance with the Multiple Spanning Tree Protocol;

10

Fig. 2, previously discussed, is a highly schematic illustration of a prior art bridged network;

Fig. 3 is a highly schematic illustration of a bridged network in accordance with a preferred embodiment of the present invention;

15

Figs. 4 and 5 are partial block diagrams of an intermediate network device in accordance with a preferred embodiment of the present invention; and

Fig. 6 is a highly schematic block diagram of a spanning tree protocol control message in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

20

Fig. 3 is a highly schematic illustration of a computer network 300 in accordance with the present invention. Network 300 includes a plurality of interconnected intermediate network devices 302a-f. Devices 302a-f are preferably bridges. Attached to the bridges 302a-f are network entities (not shown), such as local area networks (LANs) and/or end stations. The bridges 302a-f are interconnected by a plurality of segments 304, which may be shared media or point-to-point links.

25

Each bridge 302 has a plurality of ports 310 for receiving and forwarding messages across the network 300. The ports of each bridge 302, moreover, may be identified, e.g., by port numbers, such as Port 0 (P0), Port 1 (P1), Port 2 (P2), etc., so that the

entities that can be reached by a respective bridge can be associated with the particular port used to reach them. Bridge ports that are coupled directly to end stations or LANs to which no other bridge is coupled are referred to as “access ports”. Bridge ports 310 that connect to another intermediate network device, e.g., to a neighboring bridge, are referred to as “trunk ports”.

It should be understood that the network 300 of Fig. 3 is meant for illustrative purposes only, and that the present invention will operate with other networks having possibly far more complex topologies.

As shown, network 300 includes a loop. Execution of a spanning tree protocol, such as the Spanning Tree Protocol defined in IEEE Std. 802.1D-2003, prevents loops by defining a loop-free topology (i.e., an active topology) in the bridged network 300.

Furthermore, execution of the Rapid Spanning Tree Protocol (RSTP) or the Multiple Spanning Tree Protocol (MSTP), as defined at IEEE Std. 802.1Q-2003 Edition, will allow the ports 310 of the switches 302 that are part of the active topology to transition rapidly to a forwarding state so that network messages can be forwarded with minimal disruption and/or delay. However, as set forth above, in some situations, the current implementation of the RSTP and MSTP may result in certain inefficiencies. To avoid these inefficiencies, among other reasons, the bridges 302 of network 300 are configured to establish a plurality of symmetrical spanning trees within network 300.

Fig. 4 is a partial block diagram of bridge 302b. As indicated above, it has a plurality of ports 310a-c each of which is preferably identified by a number (e.g., P0-P2). One or more frame transmission and reception objects, designated generally 402, are associated with the ports 310a-c such that network messages, including data frames, received at a given port, e.g., P1, may be captured, and frames to be transmitted by bridge 302b may be delivered to a given port, e.g., P2. Frame reception and transmission objects 402 are preferably message storage structures, such as priority queues.

Bridge 302b further includes at least one protocol entity 404 that is in communicating relationship with the frame transmission and reception objects 402, and that comprises a plurality of components. In particular, the protocol entity 404 includes at least one spanning tree protocol (STP) engine 406 and one or more forwarding engines 408.

The STP engine 406 preferably comprises a plurality of subcomponents, including a port role selection state machine 410, a port transition state machine 412, and a STP control message generator 414. The STP engine 406 basically operates in compliance with the IEEE 802.1S Multiple Spanning Tree Protocol (MSTP) supplement to the IEEE Std. 802.1Q-2003, which is hereby incorporated by reference in its entirety, as extended by the novel improvement described herein. The STP engine 406 includes or is in communicating relationship with a memory 418, which may be a volatile and/or non-volatile random access memory (RAM) or some other memory device. Memory 418 is preferably organized to include a plurality of records or cells (not shown) for storing spanning tree related information or parameters, such as the bridge's numeric bridge identifier (ID), the assigned path cost for each port 310a-c, the current or "best" spanning tree information for each port P0-P2, etc.

The one or more forwarding engines 408 are preferably coupled to one or more filtering databases 420 that stores address information corresponding to the entities of network 300 (Fig. 3). Each filtering database preferably has a plurality of records (not shown) each containing a plurality of cells, including a destination address cell, a destination port cell and a corresponding timer cell. Each record in the filtering databases 420 preferably corresponds to a particular network entity.

The forwarding engines 408 are configured to switch or bridge messages, such as packets and/or frames, from a source port 310 to one or more destinations ports 310 depending on information contained in the appropriate forwarding database 420, and also depending on the spanning tree port states of the respective ports 310, as managed by STP engine 406. The forwarding engines 408 are also in communicating relationship with the STP engine 406, and may relay STP control messages, such as MSTP bridge protocol data unit (BPDU) messages, received at ports 310 thereto.

In the illustrated embodiment, bridge 302b includes transmitting and receiving circuitry, including one or more line cards and/or network interface cards (NICs) establishing ports for the exchange of network messages, one or more supervisor cards having central processing units (CPUs) and/or microprocessors and associated memory devices for performing computations and storing the results therefrom and one or more bus struc-

tures. Fig. 5 is another highly schematic, partial block diagram of bridge 302b illustrating such components. As shown in Fig. 5, bridge 302b includes a plurality of line cards 502 and 504, and a supervisor card 506. Cards 502, 504 and 506 are in communicating relationship with each other through a communication bus 508. Each of the line cards 502 and 504 includes a microprocessor (μ P) 510 and at least one memory 512. The supervisor card 506 also includes a μ P 514, as well as both a non-volatile (N-V) memory 516 and a volatile memory 518, e.g., RAM.

Referring again to Fig. 4, it will be understood by those skilled in the art that STP engine 406 and forwarding engines 408 may each comprise registers and combinational logic configured and arranged to produce sequential logic circuits. In the illustrated embodiment, engines 406 and 408 are preferably software modules or libraries containing program instructions pertaining to the methods described herein and executable by one or more processing elements, such as the microprocessors 510 and/or 514 (Fig. 5), of bridge 302b. Other computer readable media may also be used to store and execute these program instructions. Nonetheless, those skilled in the art will recognize that various combinations of software and hardware, including firmware, may be utilized to implement the present invention.

Suitable intermediate network device platforms for use with the present invention include, but are not limited to, the commercially available series of switches from Cisco Systems, Inc. of San Jose, California, such as the Catalyst 4000 and 6000.

As set forth herein, bridges 302 of network 300 are configured to create a plurality of spanning tree instances through network 300. In the illustrative embodiment, they create one spanning tree instance per bridge where each bridge is the root of one spanning tree instance. Each bridge 302 is also configured to forward messages that it received on one of its access ports along the spanning tree instance for which the bridge is the root. In this way, messages traversing network 300 follow more efficient paths.

Because a bridge typically learns which port to use in order to reach another device within a network based upon the bridge port at which frames from that device have been received by it, the active topology defined by each pair of bridges needs to be symmetric. Otherwise a loss of communication might result. Accordingly, the bridges 302

are further configured such that the two spanning tree instances associated with each pair of bridges is symmetric. A pair of spanning tree instances are considered symmetric when the route followed by a message sourced by either of the two bridges is the same, i.e., the two messages pass through the same set of intermediary bridges but in the reverse order from each other.

To ensure that each pair of spanning tree instances is symmetric, the bridges 302 of network 300 are configured, first such that the path cost for a given segment is the same for all bridges coupled to that segment, and second to exchange and process novel STP control messages. In one embodiment, a network administrator manually configures the bridges 302 of network 300 such that all bridges coupled to a given network segment have the same path cost assigned to that segment. Those skilled in the art will understand that other techniques besides manual configuration may be used. For example, an information distribution protocol may be used to convey path costs.

A conventional MSTP BPDU message 100 (Fig. 1) includes a plurality of MSTI Configuration messages 152; one for each spanning tree instance for which the MSTP BPDU 100 is carrying information. In accordance with the present invention, the novel STP control message is formed by modifying each MSTI Configuration message to carry path-tracking or recording information. As the STP control message is propagated across the network 300, it is updated to reflect each bridge through which it is being propagated.

Fig. 6 is a highly schematic illustration of a preferred format of a STP control message 600. The STP control message 600 includes a header 602 that is preferably compatible with the Media Access Control (MAC) layer of the respective LAN standard, e.g., Ethernet, and corresponds to the header 102 (Fig. 1) of a MSTP BPDU 100. Appended to the header 602 is an "outer" part 604, which has the same format as outer part 104 (Fig. 1) of the MSTP BPDU 100. That is, outer part 604 corresponds to an RSTP BPDU message and is recognized as a valid RSTP BPDU message by bridges that do not implement MSTP or the Spanning Tree Routing extension to MSTP. The STP control message 600 further includes an "inner" part 606 that is utilized by bridges 302 to establish the IST and the symmetrical MSTIs. Inner part 606 preferably includes a version 1 length field 608, a version 3 length field 610, a MST Configuration ID field 612, a CIST

Internal Root Path Cost field 614, a CIST Bridge ID field 616, a CIST Remaining Hops field 618, and a plurality of MSTI Configuration Messages 620.

As further shown in Fig. 6, each MSTI Configuration Message 620 preferably includes a MSTI flags field 622, a MSTI Regional Root ID field 624, a MSTI Internal Root Path Cost field 626, a Path Tracking field 628, a MSTI Bridge Priority field 630, a MSTI Port Priority field 632, and a MSTI Remaining Hops field 634.

With reference to Fig. 3, bridge 302a generates a STP control message 600 message for transmission from its ports. Within the STP control message 600, bridge 302a inserts a MSTI Configuration message 620 for the spanning tree instance for which bridge 302a is the root. Bridge 302a loads the MSTI flags field 622 of this MSTI Configuration message 620 as specified by IEEE 802.1Q. Since this MSTI Configuration message 620 corresponds to the spanning tree instance for which bridge 302a is the root, it enters its own Bridge ID value in the MSTI regional root ID field 624, and sets the MSTI internal root path cost field 626 to null or zero. In accordance with the present invention, bridge 302a also sets the path tracking field 628 to null or zero, since this MSTI Configuration message 600 is being generated by bridge 302a, and has yet to pass through any bridges 302. Fields 630-634 are preferably loaded in accordance with the conventional MSTP. Bridge 302a then transmits the STP control message 600 from its ports 310.

One or more of the STP control messages from bridge 302a is received by bridge 302c and by bridge 302d. Bridge 302c replaces the current value contained in the root path cost field 626 with the sum of the current value and the cost of the path between it and bridge 302a. Bridge 302c also updates the value of path tracking field 628 to indicate that this STP control message 600 has been propagated through bridge 302c. In one embodiment, bridge 302c updates the path-tracking field 628 as follows. First, bridge 302c applies a hash function to its Bridge ID value so as to generate a hashed Bridge ID value. A Bridge ID is typically 64-bits long. The first 4-bits are a settable priority component, the next 12-bits specify the spanning tree instance, and the last 48-bits is the bridge's globally unique MAC address, e.g., the MAC address for port P0. The hash function may be applied to the entire 64-bit Bridge ID. Alternatively, it may be applied

to just a portion of the Bridge ID, such as the 48-bit MAC address or some other field or combination of fields.

In a preferred embodiment, the hash function produces a value that is less than 64-bits. Nonetheless, in an alternative embodiment the hash function may produce an
5 output that is greater than 64-bits.

After generating the hashed Bridge ID value, bridge 302c replaces the current value of the path-tracking field 628 with the sum of the hashed value and the path tracking field's current value. Bridge 302c preferably modifies the values of fields 630-634 as specified by IEEE Std. 802.1Q-2003. Bridge 302c then propagates the as-modified STP
10 control message 600 from its ports 310.

Those skilled in the art will understand that any number of well-known hash functions may be employed, such as Cyclic Redundancy Check (CRC), Secure Hash Algorithm (SHA), Message Digest Algorithm 4 or 5 (MD4 or MD5), Race Integrity Primitives Evaluation Message Digest (RIPEMD).

15 Bridge 302d similarly modifies the STP control message 600 received from bridge 302a before propagating it from its own ports 310. Bridge 302d, however, updates the path tracking field by hashing its Bridge ID and then adding this hashed Bridge ID value to the current path tracking value in order to create the new path tracking value for field 628. This process is repeated by bridges 302e and 302f, which update the path
20 tracking field 628 of the STP control messages 600 received by them. In particular, bridge 302e updates the path-tracking field 628 by hashing its Bridge ID and adding the hashed value to the current path tracking value. Bridge 302f similarly updates the path-tracking field 628 by hashing its Bridge ID and adding the hashed value to the current path tracking value.

25 Bridge 302b receives two MST Configuration messages 620 for the spanning tree instance for which bridge 302a is the root. The first MST Configuration message 620 is received from bridge 302e, and has a value in its path tracking field 628 that reflects that this MST Configuration message 620 was propagated through bridges 302d and 302e. The second MST Configuration message 620 is received from bridge 302f. It carries a
30 value in its path tracking field 628 that reflects that it was propagated through bridges

302c and 302f. Bridge 302b utilizes the values from these two MST Configuration messages to select its root port for the spanning tree instance for which bridge 302a is the root. First, bridge 302b determines whether one of the received MST Configuration messages 620 specifies a lower cost path to the root, i.e., to bridge 302a, than the other. If so, bridge 302b selects as its root port, the port providing the lowest cost path to the root, i.e., to bridge 302a. Suppose the path costs specified by both MSTI Configuration messages 620 are the same. Under the conventional MSTP operation, bridge 302b would then look to the Bridge IDs of the bridges that sourced the two MSTI Configuration messages as carried in field 610 to elect its root port, i.e., the Bridge IDs of bridges 302e and 302f.

Instead of looking to the source Bridge IDs, however, the STP engine 406 of bridge 302b is configured to utilize the values of the path-tracking field 628 to elect its root port. That is, with the present invention, in situations where the root path costs for two or more ports are the same, the STP engine 406 elects, as its root port, the port on which the MSTI Configuration message 620 with the lowest path tracking value was received. Suppose that the MSTI Configuration message 620 from bridge 302e has a lower path tracking value than the path tracking value carried by the MST configuration message 620 from bridge 302f. Bridge 302b responds by electing its port coupled to bridge 302e as the root port for the spanning tree instance for which bridge 302a is the root. In response, the port role selection state machine 410 assigns the port to the Root Port Role, and the port transition state machine transitions the port to the forwarding state. The port coupled to bridge 302f is assigned to the Alternate Port Role, and is transitioned to the blocking state, as indicated by the "X".

Those skilled in the art will understand that, rather than using lowest path tracking value, the STP engines of bridges 302a-f, could alternatively be configured to use the highest path tracking values for electing root ports. Similarly, those skilled in the art will understand that a different function, other than summing, may be used in updating the path-tracking record field.

Bridge 302a similarly receives two MSTI Configuration messages 620 originally generated by bridge 302b for the spanning tree instance for which bridge 302b is the root. The first such MSTI Configuration message carries a value in its path tracking field

628 that indicates it was propagated through bridges 302f and 302c, while the second MSTI Configuration message 620 carries a value in its path tracking field 628 that indicates it was propagated through bridges 302e and 302d. As was the case with bridge 302b, bridge 302a also relies on the values contained in these two path-tracking fields when electing its root port for the spanning tree instance for which bridge 302b is the root. Bridge 302a looks to the values carried by the path tracking fields 628, because the root path costs for its two ports 310 leading to bridge 302b are the same.

Significantly, the value contained in the path tracking field 628 of the MSTI Configuration message 620 received from bridge 302d is lower than the path tracking field 628 of the MSTI Configuration message 620 received from bridge 302c. This is true because the path tracking value is the same regardless of the direction taken by the respective STP control message. That is, (hash of Bridge ID for bridge 302d) + (hash of Bridge ID for bridge 302e) equals (hash of Bridge ID for bridge 302e) + (hash of Bridge ID for bridge 302d). Similarly, (hash of Bridge ID for bridge 302c) + (hash of Bridge ID for bridge 302f) equals (hash of Bridge ID for bridge 302f) + (hash of Bridge ID for bridge 302c). Accordingly, bridge 302a elects, as its root port for the spanning tree instance for which bridge 302b is the root, its port coupled to bridge 302d. This port is thus assigned to the Root Port Role, and transitioned to the forwarding state. The port coupled to bridge 302c is assigned to the Alternate Port Role, and transitioned to the blocking state, as indicated by the "X".

As shown, utilization of the path-tracking feature of the present invention ensures that the spanning trees rooted at any pair of bridges 302 of network 300 will be symmetrical.

In the preferred embodiment, network 300 is further configured such that each spanning tree instance is assigned a corresponding VLAN ID (VID). When a bridge 302 receives a message on one of its access ports, it preferably tags the message with the VID assigned to the spanning tree instance for which the bridge is the root. In this way, the message is routed along the spanning tree instance for which the bridge that initially received the message is the root. Those skilled in the art will understand that the VLAN configuration may be performed manually by a network administrator.

It should be understood that, rather than applying a hash function, the bridges 302 could alternatively apply some function, such as summing, to their full Bridge ID values in order to update the path-tracking field 628. It should also be understood that a value other than Bridge IDs could be chosen as the basis of the path record. Furthermore, other approaches besides hash functions may be employed. Such other approaches preferably take a value associated with each bridge, which may be its Bridge ID as described here or some other value, and use that selected value as a key to generate a random number. In a preferred embodiment, the selected bridge value, and the applied function(s) are chosen so as to minimize, if not eliminate, the chance of collisions, i.e., the same path-tracking record value being produced even though two different paths are traversed.

In a preferred embodiment, the path-tracking field is configured to have a bit for each bridge 302 of network 300. Such a configuration is shown in the alternative path-tracking field 628a of Fig. 6, in which field 628a includes six bits; one for each bridge 302. In this embodiment, before transmitting a MSTI Configuration message 620, a root bridge, e.g., bridge 302a, asserts, e.g., sets to "1", the bit of field 628a that is associated with the root bridge, and de-asserts, e.g., sets to zero or null, all other bits of field 628a. As each downstream bridge, e.g., bridge 302d, receives the STP control message 600 carrying path tracking field 628a, it asserts, e.g., sets to "1", the bit of field 628a that is assigned to it. Thus, for a path tracking field originally sourced by bridge 302a and propagating through bridges 302d and 302e, the bits of the path tracking field 628a corresponding to these three bridges will be asserted, while the other bit positions will all be de-asserted. Again, when the root path costs for two or more ports are the same, a bridge 302 elects as its root port, the port on which the lower (or, alternatively the higher) path tracking value was received.

The STP engine 406 of each bridge 302 may be configured through network administrator action with information indicating which bit of the path-tracking field 628a is assigned to that bridge 302. Alternatively, this information could be provided to the bridges 302 through execution of an information distribution protocol.

Such a configuration of the path-tracking field 628a avoids situations in which the hashed sum of two different groups of Bridge IDs nonetheless comes to the same value, thereby ensuring deterministic behavior.

The present invention can also be used with other techniques for establishing symmetrical spanning trees. For example, U.S. Pat. Application Ser. No. 11/182,564, filed July 14, 2005 entitled *Methods and Devices for Improving the Multiple Spanning Tree Protocol*, commonly owned with the present application, and which is hereby incorporated by reference in its entirety, discloses a technique for creating symmetrical spanning trees using a Reflection Vector.

The Reflection Vector includes a separate bit for each spanning tree instance being defined in the network. Each MSTI Configuration message, moreover, is configured to carry a corresponding Reflection Vector. When a root bridge generates the MSTI Configuration message for the spanning tree instance for which it is the root, it initially asserts all of the bits of the Reflection Vector, e.g., it sets them all to "1". The root bridge then checks to see if the given port from which the Reflection Vector is being sent is not the root port for any of the spanning tree instances. If the given port is not the root port for a spanning tree instance, the bridge deasserts the bit position of the Reflection Vector corresponding to the spanning tree instance. Downstream bridges similarly deassert those bit positions of the Reflection Vector corresponding to spanning tree instances for which the port from which the Reflection Vector is being sent is not the root port. For those spanning tree instances for which the given port is the root port, then the corresponding bits of the Reflection Vector are left "as is", i.e., as received by the downstream bridge.

In a preferred embodiment, the path tracking feature of the present invention is combined with the Reflection Vector of the incorporated patent application. That is, the MST Configuration messages are modified to convey both path-tracking fields and Reflection Vectors. In a further embodiment, the Reflection Vector is modified to include only the bits for those spanning tree instances that are inferior to the spanning tree instance associated with the Reflection Vector. In particular, a unique number can be attributed to each spanning tree instance. When a given bridge builds a reflection vector

for the spanning tree instance for which it is the root, it need only include the bits for those spanning tree instances whose unique value is less than the spanning tree instance for which it is the root.

By incorporating the path tracking feature of the present invention with the Reflection Vector, a Reflection Vector is able to converge in a single pass across the network.

The foregoing description has been directed to specific embodiments of this invention. It will be apparent, however, that other variations and modifications may be made to the described embodiments, with the attainment of some or all of their advantages. For example, rather than forming a separate spanning tree instance for each bridge in the network, the bridges may be organized into groups and symmetric spanning trees established for each pair of groups. Therefore, it is an object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:

CLAIMS

1 1. In an intermediate network device having a plurality of ports for receiving and
2 forwarding messages within a computer network, a method of forming a plurality of sym-
3 metric spanning tree instances, the method comprising:

4 receiving a Spanning Tree Protocol (STP) control message, wherein the received
5 STP control message includes a path-tracking record field corresponding to a given span-
6 ning tree instance, the path-tracking record field having information indicating which
7 other intermediate network devices have propagated the STP control message; and
8 utilizing the path-tracking record field in computing an active topology for the
9 given spanning tree instance.

1 2. The method of claim 1 wherein the STP control message includes a plurality of
2 Multiple Spanning Tree (MST) configuration messages, each MST configuration mes-
3 sage corresponding to a different spanning tree instance and having a respective path-
4 tracking record field.

1 3. The method of claim 1 further comprising:
2 updating the path-tracking record field of the received STP control message by
3 incorporating information indicating that the STP control message was received by the
4 intermediate network device; and
5 sending the STP control message with the updated path-tracking record field from
6 at least one port of the intermediate network device.

1 4. The method of claim 3 wherein the intermediate network device has an identi-
2 fier, and the updating comprises:
3 computing an output from the identifier of the device;
4 applying a function to the output and a current value of the path-tracking record
5 field in order to generate a new value; and
6 entering the new value in the path-tracking record field.

1 5. The method of claim 4 wherein
2 the identifier is a Bridge Identifier (ID), and
3 the output is generated by hashing the Bridge ID.

1 6. The method of claim 5 wherein the function is a summing function.

1 7. The method of claim 1 further comprising:
2 receiving two or more STP control messages on corresponding ports, each re-
3 ceived STP control message having a respective path-tracking record field and a root path
4 cost for the given spanning tree instance, wherein the root path costs of the two or more
5 received STP control messages are the same; and,
6 electing as a root port for the given spanning tree instance, the port on which a
7 best path-tracking record field was received.

1 8. The method of claim 7 wherein a lower path-tracking record field is better than
2 a higher path-tracking record field.

1 9. The method of claim 7 wherein a higher path-tracking record field is better than
2 a lower path-tracking record field.

1 10. The method of claim 1 wherein
2 the network includes a plurality of intermediate network devices, and
3 the plurality of intermediate network devices cooperate to establish a plurality of
4 spanning tree instances such that each intermediate network device is a root of a respec-
5 tive spanning tree instance.

1 11. The method of claim 10 wherein the path-tracking record field includes a bit
2 assigned to each of the plurality of intermediate network devices, the method further
3 comprising:

4 updating the path-tracking record field of the received STP control message by
5 asserting the bit assigned to the intermediate network device that received the STP con-
6 trol message; and

7 sending the STP control message with the updated path-tracking record field from
8 at least one port of the intermediate network device.

1 12. An intermediate network device configured to receive and forward messages
2 within a computer network, the device having a plurality of ports for connecting the de-
3 vice to one or more network entities and other devices, the intermediate network device
4 comprising:

5 a spanning tree protocol (STP) engine configured and arranged to cooperate in the
6 creation of a plurality of spanning tree instances in the network, the STP engine having a
7 control message generator, wherein

8 the control message generator is configured to create a Multiple Spanning Tree
9 Instance (MSTI) configuration message for each of the plurality of spanning tree in-
10 stances, and to insert within each such MSTI configuration message a corresponding
11 path-tracking record field, and

12 the STP engine uses the path-tracking record field of a received STP control mes-
13 sage in creating the corresponding spanning tree instance.

1 13. The intermediate network device of claim 12 wherein the STP engine, in re-
2 sponse to receiving two or more STP control messages on corresponding ports that, for a
3 given spanning tree instance, have respective path-tracking record fields and specify the
4 same root path cost for the given spanning tree instance, elects as a root port for the given
5 spanning tree instance, the port on which a best path-tracking record field was received.

1 14. The intermediate network device of claim 12 further comprises a Bridge Iden-
2 tifier (ID) value, wherein the STP engine

3 computes a hash value of the Bridge ID value,

4 applies a function to the computed hash value and a current value of the path-
5 tracking record field to generate a new value; and

6 enters the new value in the path-tracking record field.

1 15. The intermediate network device of claim 14 wherein the function is a sum-
2 ming function.

1 16. The intermediate network device of claim 12 wherein
2 the device is the root of a given spanning tree instance,
3 a Virtual Local Area Network (VLAN) Identifier (VID) is assigned to the span-
4 ning tree instance for which the device is the root, and
5 the device tags a message received by the device from a network entity directly
6 coupled thereto with the VID before forwarding the message into the network.

1 17. A data structure stored in a memory for use by an intermediate network device
2 in computing an active topology of a spanning tree instance in a computer network hav-
3 ing a root device, the data structure comprising:
4 an identifier of an intermediate network device considered to be the root of the
5 spanning tree instance;
6 a cost assigned to a path leading to the root device; and
7 a path-tracking record indicating one or more other devices between the device
8 considered to be the root of the spanning tree instance and a current device.

1 18. The data structure of claim 17 wherein
2 the computer network has a plurality of other intermediate network devices,
3 each other device has a Bridge Identifier (ID), and
4 the path-tracking record is derived from the Bridge IDs of the one or more other
5 devices between the root device and the current intermediate network device.

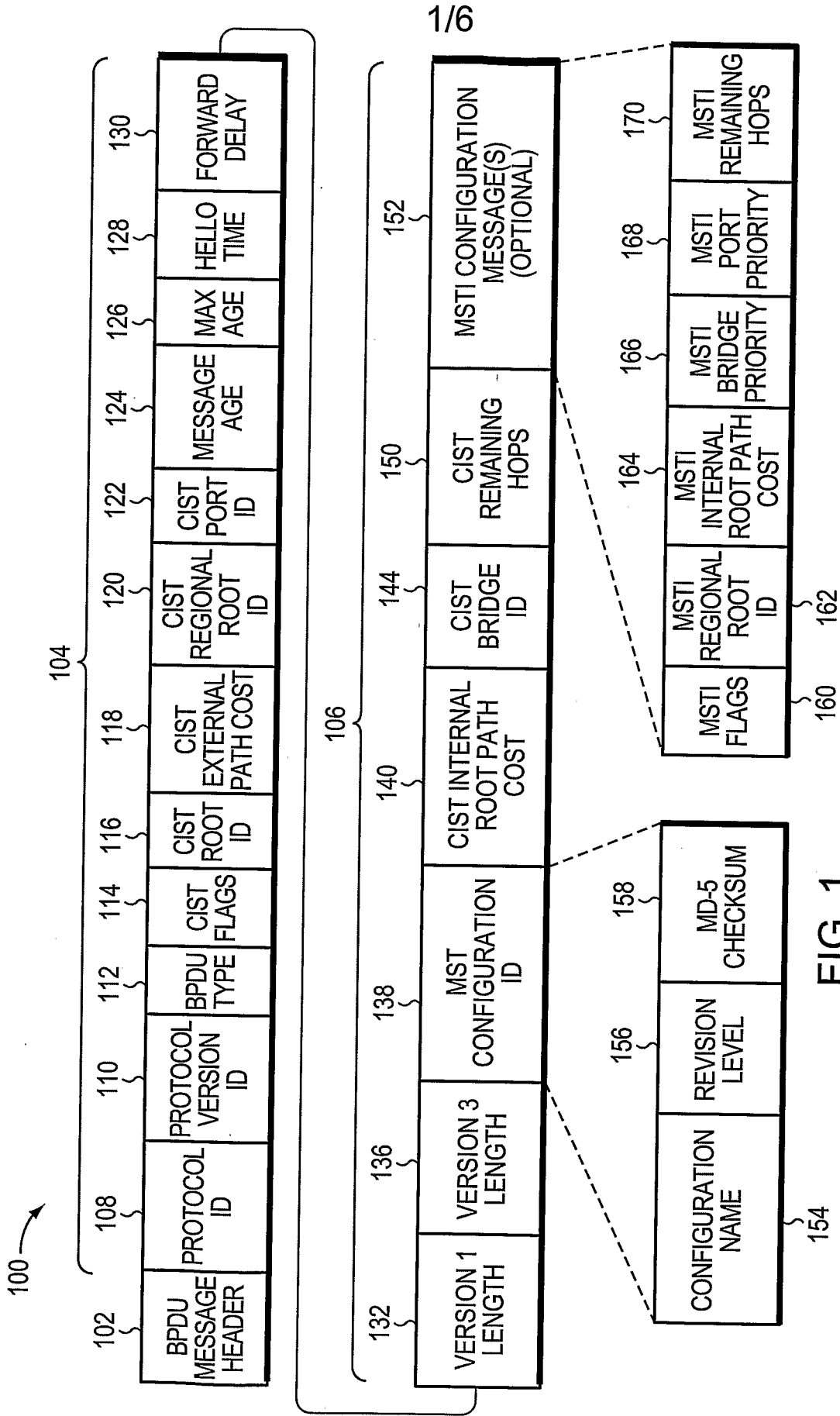


FIG. 1
(PRIOR ART)

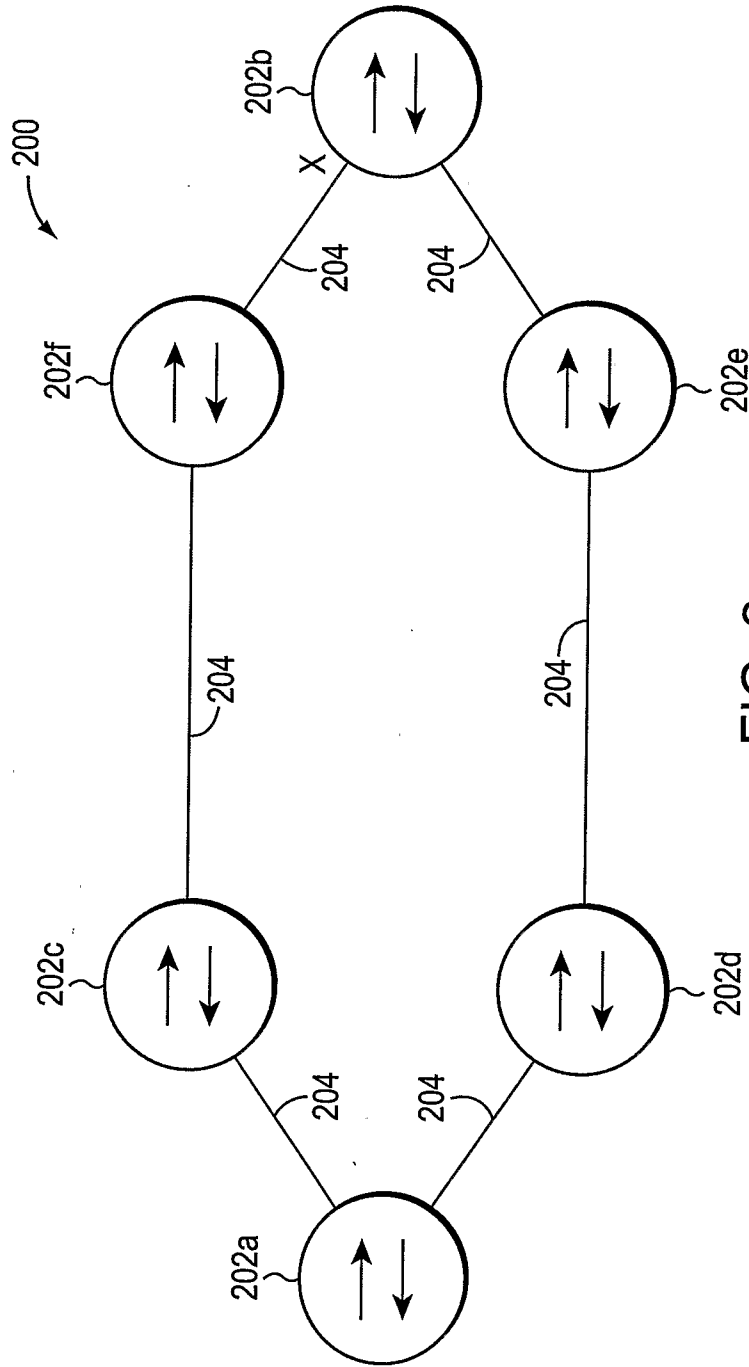


FIG. 2
(PRIOR ART)

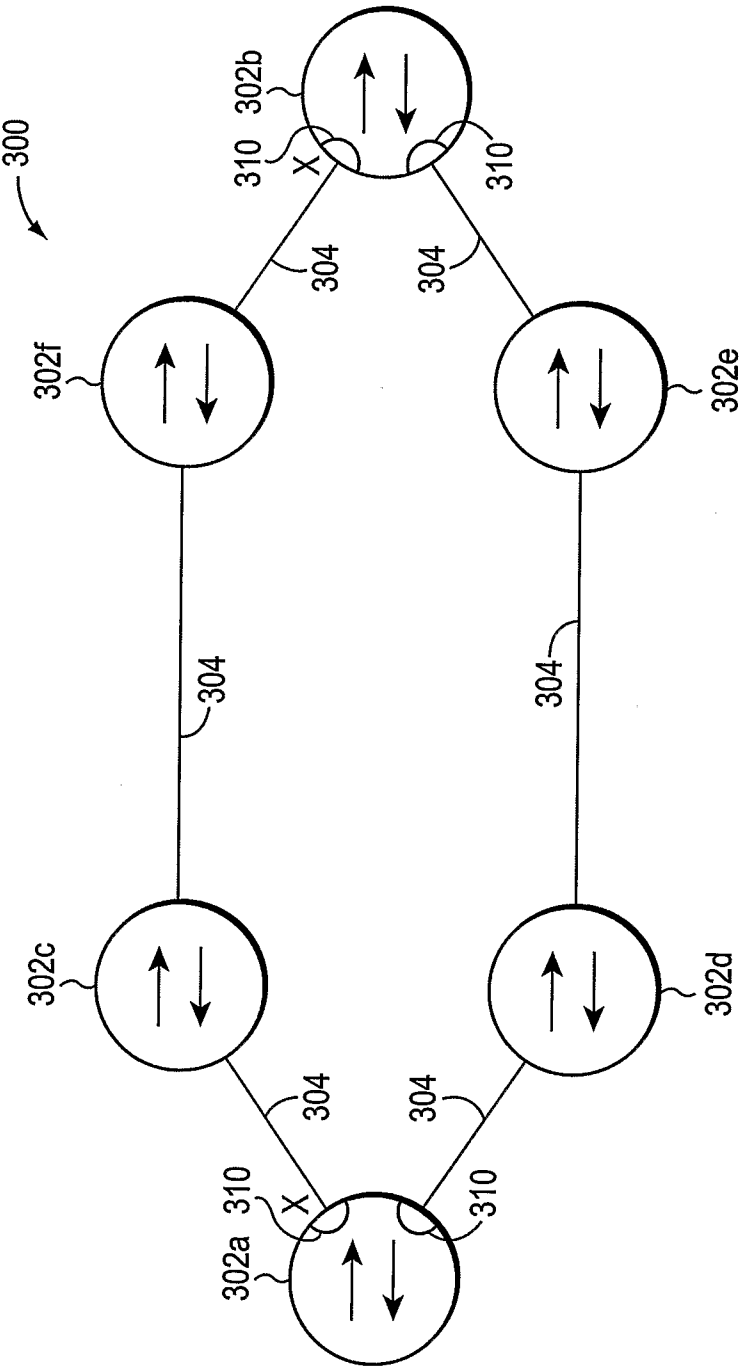


FIG. 3

4/6

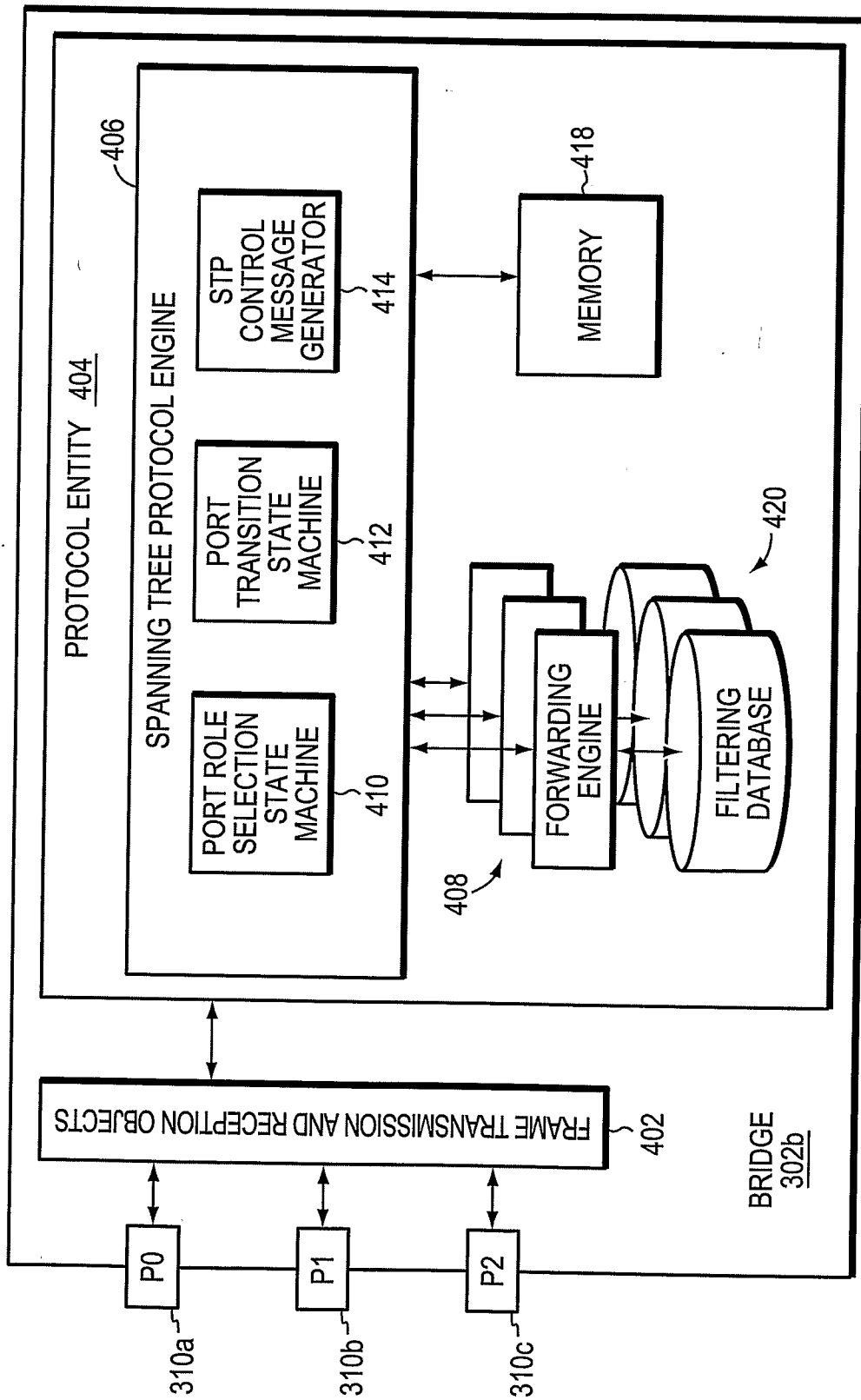


FIG. 4

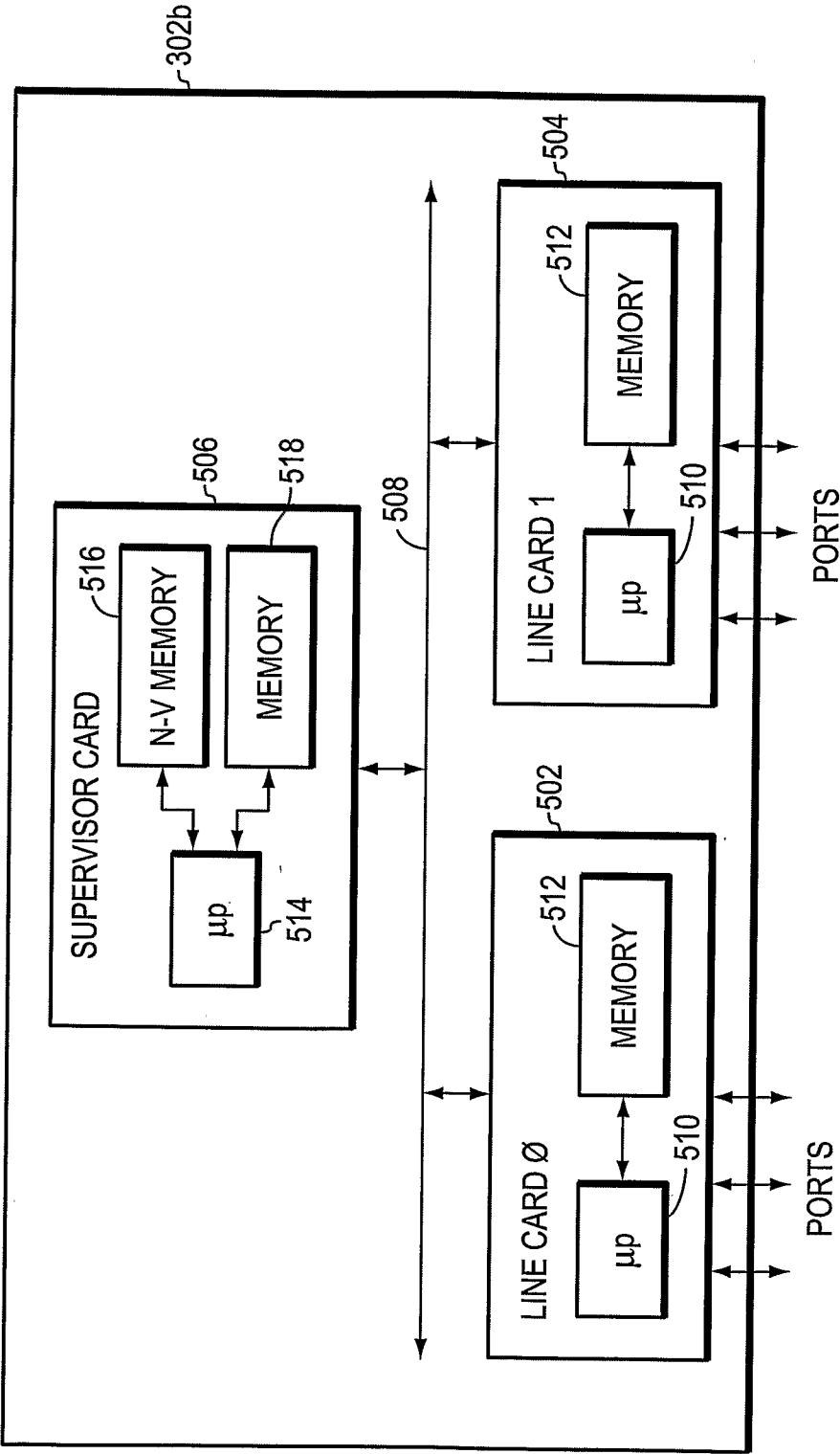


FIG. 5

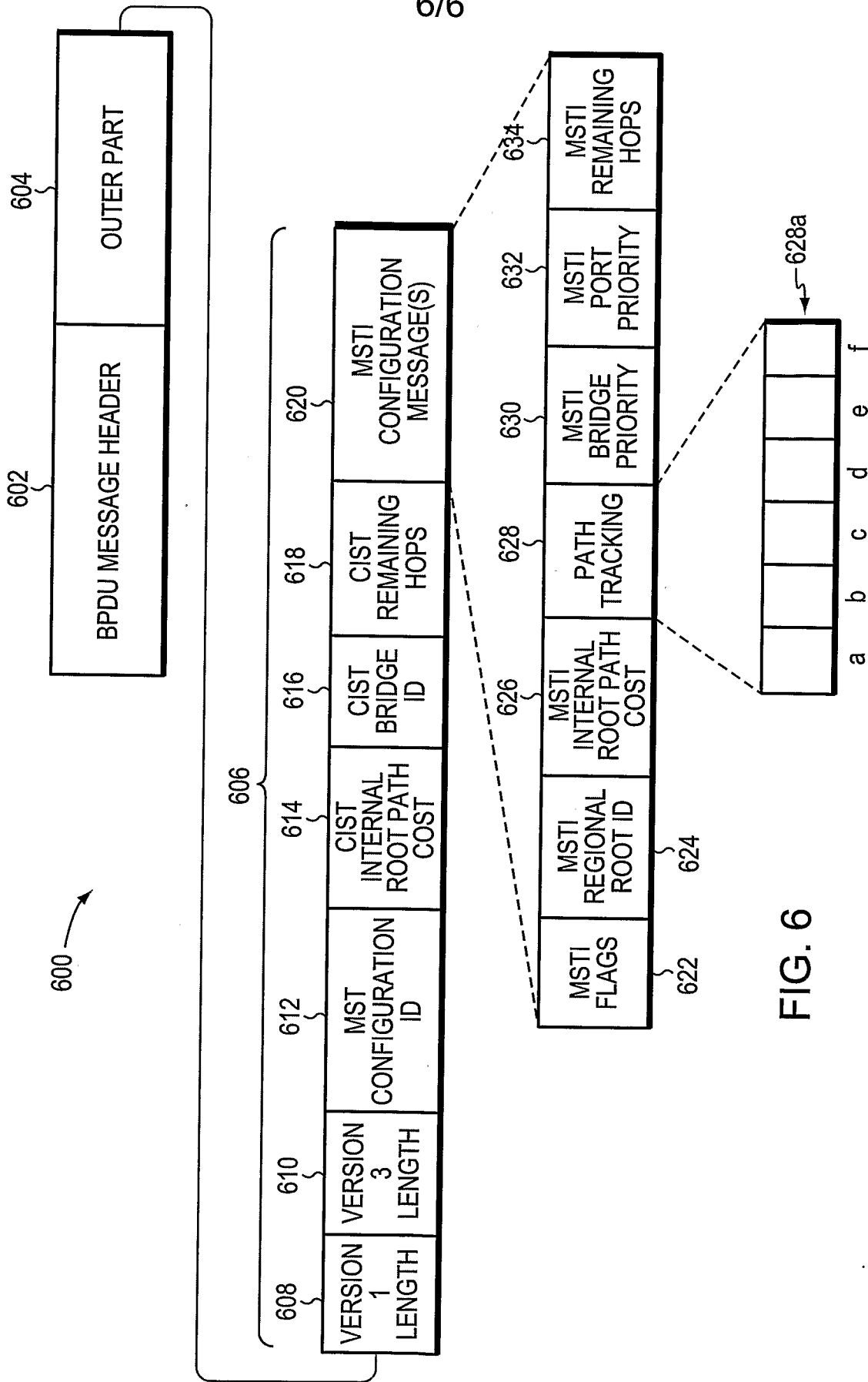


FIG. 6