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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 June 2004 (17.06.2004)

PCT

(10) International Publication Number
WO 2004/051938 A2

(51) International Patent Classification[7]: H04L 12/46

(21) International Application Number:
PCT/US2003/036452

(22) International Filing Date:
12 November 2003 (12.11.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/429,897 27 November 2002 (27.11.2002) US
10/430,491 5 May 2003 (05.05.2003) US

(71) Applicant: **ANDIAMO SYSTEMS, INC.** [US/US]; 375 East Tasman Drive, San Jose, CA 95134 (US).

(72) Inventors: **DESAI, Tushar**; 1110 Polynesia Drive, #214, Foster City, CA 94404 (US). **GUPTA, Shashank**; 430 Oak Grove Drive, #207, Santa Clara, CA 95054 (US). **JAIN, Praveen**; 4684 San Lucas Way, San Jose, CA 95135 (US). **GHOSH, Kalyan, K.**; 2281 Esperanca Avenue, Santa Clara, CA 95054 (US).

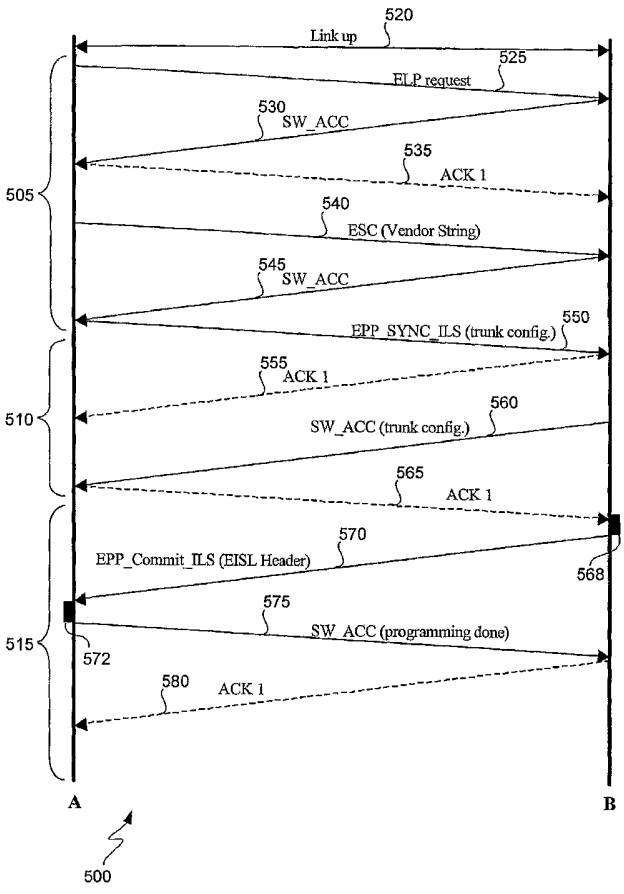
(74) Agent: **SAMPSON, Roger, S.**; Beyer Weaver & Thomas, LLP, P.O. Box 778, Berkeley, CA 94704-0778 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: METHODS AND DEVICES FOR EXCHANGING PEER PARAMETERS BETWEEN NETWORK DEVICES



(57) Abstract: Methods and devices are provided for detecting whether peer ports internconnecting two network devices can perform a novel protocol called Exchange Peer Parameters ("EPP"). If the peer ports are so configured to perform EPP, EPP services are exchanged between the peer ports. In a first phase, information is exchanged about peer port confirgurations of interest. In a second phase, the results of the exchanged of information are applied to hardware and/or software of the respective ports, as needed.

WO 2004/051938 A2

European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (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.

METHODS AND DEVICES FOR EXCHANGING PEER PARAMETERS BETWEEN NETWORK DEVICES

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to data networks. More specifically, the invention relates to the configuration of routers, switches and other network devices within such data networks.

2. Description of Related Art

Several limitations may be encountered when configuring networks such as local area networks, storage area networks and the like. There are a variety of network devices, such as routers, switches, bridges, etc., which may be used to configure such networks. Some of these network devices have greater capabilities than others. For example, some devices may readily be configured to support logical networks superimposed upon a physical network (e.g., virtual local area networks ("VLANs") or virtual storage area networks ("VSANs")) and some may not.

In order to allow multiple VLANs to share a single inter-switch link on the underlying physical topology, the interswitch link protocol ("ISL") was developed at Cisco Systems. See for example U.S. Pat. No. 5,742,604, entitled "Interswitch link mechanism for connecting high-performance network switches," Edsall, et al., issued on April 21, 1998 to Cisco Systems, Inc., which is hereby incorporated by reference for all purposes. ISL provides an encapsulation mechanism for transporting packets between ports of different switches in a network on the basis of VLAN associations among those ports

In one example, it would be useful to transport packets of different frame types using the same inter-switch link instead of dedicating inter-switch links for different frame types. For example, it would be desirable if links between network devices could carry both Ethernet and Fiber Channel ("FC") frames.

It is also important to determine as quickly as possible whether a network device has certain capabilities. For example, it would be very useful to determine quickly whether a peer port of another network device is configured (or could be configured) to carry frames of particular VLANs or VSANs, and to configure the network device as needed. Otherwise, various problems (including dropped frames) will ensue if the network device is connected to other devices that are transmitting

frames for the wrong VLAN or VSAN. However, testing and configuring network devices for such capabilities can be time-consuming.

SUMMARY OF THE INVENTION

According to some aspects of the invention, a new protocol, known herein as Exchange Peer Parameters ("EPP"), is provided for communication between peer ports of network devices that form part of the fabric of a network. In some embodiments, EPP protocol is used to exchange information and/or to configure E or F ports of an FC network.

Methods and devices are provided for detecting whether an attached peer port of a network device can exchange peer parameters with the corresponding port according to a novel Exchange Peer Parameters ("EPP") protocol. If the peer port is so configured, EPP service exchanges are performed with the peer port. In a first phase, information is exchanged about peer port configurations of interest. In a second phase, the results of the exchange of information are applied to hardware and/or software of the peer ports, as needed.

According to some aspects of the invention, when an inter-switch link is formed, a port of a peer network device is interrogated to determine whether it can support EPP protocol. If so, EPP service exchanges are performed with the peer port.

According to other aspects of the invention, configuration information is exchanged between peer ports in a network after an inter-switch link has been formed between the peer ports and after data frames have been transmitted to and from the peer network device. Such an information exchange may occur, for example, when the trunk mode of one of the ports has been changed during operation of the port. The results of the exchange of information are applied to hardware and/or software of the peer ports, as needed.

According to some implementations of the invention, methods and devices are provided for configuring a port of a network device in trunking mode so that all frames are transmitted in a novel format known as extended inter-switch link ("EISL") format, which will be discussed in more detail below. According to some such aspects of the invention, when an inter-switch link is formed, a port of a peer network device is interrogated to determine whether it can be a trunking port. If so, the port is configured to be in trunking mode using the EPP protocol.

According to some preferred aspects of the invention, the EPP protocol is used after the Exchange Switch Capabilities ("ESC") protocol. ESC may be used to

exchange a set of protocols supported by the switch. EPP is one such protocol in the set of protocols. The EPP protocol is used, for example, to determine whether a port of a network device is configurable for supporting VLANs, VSANs and/or EISL. The EPP protocol can be used, for example, to configure an E or F port for EISL. If an E port is so configured, the port is referred to as a "trunking E port" or a TE port.

According to some implementations of the invention, a method is provided for modifying configurations of peer ports interconnecting network devices. The method includes: determining that the interconnected peer ports, comprising a first port of a first network device and a second port of a second network device, can support Exchange Peer Parameters protocol; exchanging configuration information using the Exchange Peer Parameters protocol between the interconnected peer ports; and configuring the interconnected peer ports according to the exchanged information.

The determining step can involve exchanging information between the first port and the second port via, for example, Exchange Link Parameter protocol or Exchange Switch Capability protocol. The exchanging step can involve exchanging frames in, for example, type-length-value format or a fixed frame length format. The configuration information can include, for example, virtual storage area network information or trunk mode information. The configuration information can be exchanged when the interconnected peer ports are being initialized or when the interconnected peer ports have already been initialized. The configuration step can include configuring the hardware and/or the software of the interconnected peer ports according to the exchanged information.

Alternative implementations of the invention provide a method for modifying a configuration of a network device. The method includes: determining that a first expansion port of a first network device, the first expansion port attached to a second expansion port of a second network device, can be configured to transmit frames in Extended Interswitch Link format; and configuring the first expansion port to transmit frames in Extended Interswitch Link format.

The determining step can include exchanging trunk mode information between the first expansion port and the second expansion port via Exchange Peer Parameters protocol. The configuring step can include configuring the hardware and/or software of the first expansion port to enable transmission of frames in Extended Interswitch Link format. The configuring step can involve informing the second expansion port via Exchange Peer Parameters protocol that the configurations have been applied to the first expansion port.

Some embodiments of the invention provide a computer program for causing a first expansion port of a first network device to modify a configuration of a second expansion port of a second network device. The computer program causes the first expansion port to perform the following steps: determining that the second expansion port can be configured as a trunking port for transmitting frames in Extended Interswitch Link format; and configuring the second expansion port as a trunking port.

The determining step may involve exchanging information between the first expansion port and the second expansion port via Exchange Link Parameter protocol or via Exchange Switch Capability protocol. The configuring step can include exchanging information between the first expansion port and the second expansion port via Exchange Peer Protocol.

Alternative aspects of the invention provide a carrier wave embodying an encoded data signal for modifying a configuration of a network device. The encoded data signal includes: a command code field for identifying whether a command is from a synchronization phase or a commit phase of a process for configuring an expansion port of the network device; and a command identifier field for indicating whether a request to perform part of the process has been accepted or rejected.

The encoded data signal may also include trunk configuration information. The trunk configuration information can include, e.g., administratively configured trunk mode information for trunk mode negotiation, virtual storage area network list information, or port virtual storage area network information. The administratively configured trunk mode information can include a setting selected from the group consisting of ON, OFF and AUTO.

Yet other embodiments of the invention provide an apparatus for modifying a configuration of a network device. The apparatus includes: a mechanism for determining that the interconnected peer ports, comprising a first port of a first network device and a second port of a second network device, can support Exchange Peer Parameters protocol; a mechanism for exchanging configuration information using the Exchange Peer Parameters protocol between the interconnected peer ports; and a mechanism for configuring the interconnected peer ports according to the exchanged information. These mechanisms may or may not be separate devices, according to the implementation.

Still other embodiments of the invention provide a first network device for modifying a configuration of a second network device. The first network device is configured to perform the following steps: determining that a port of the second

network device can support Exchange Peer Parameter protocol; and causing the port to be configured based on configuration information exchanged between the first network device and the port via Exchange Peer Parameters protocol.

The determining step can include exchanging information between the first network device and the port via Exchange Link Parameter protocol or Exchange Switch Capability protocol. The configuring step can include exchanging information between the first network device and the port via Exchange Peer Parameter protocol.

A further understanding of the nature and advantages of the present invention may be realized by reference to the remaining portions of the specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a storage area network.

Fig. 2 depicts an EISL frame.

Fig. 3 illustrates a simplified frame having an EISL header.

Fig. 4 illustrates an exemplary stack for implementing an exchange peer protocol ("EPP").

Fig. 5 is a flow diagram that outlines the processes of determining that a device can be configured for EPP and implementing EPP.

Fig. 5A is a diagram of a time-length-value frame.

Fig. 6 is a table that indicates how differences are resolved between a local trunk mode and a peer trunk mode.

Fig. 7 is a diagram that indicates VSAN bit map information from port A and port B and the resulting VSAN intersection bit map.

Fig. 7A is a flow chart that outlines a process for implementing the EPP SYNC and commit phases after a link has previously been established.

Fig. 8 is a flow chart that outlines the EPP process for an initiating port.

Fig. 9 is a flow chart that outlines the EPP process for a receiving port.

Fig. 10 is a table that describes one example of an EPP header.

Fig. 11 depicts a network device that may be configured to perform the methods of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 indicates network 100, which is a storage area network ("SAN") according to some preferred aspects of the present invention. Although the following description will focus on SANs and their corresponding protocols, etc., the present invention is applicable to other networks, such as LANs.

SAN 100 includes nodes 105 and 110, which may be host devices such as personal computers. SAN 100 also includes nodes 115, 120 and 125, which are storage devices in this instance. Although Internet 130 is not part of SAN 100, it is connected to SAN 100 via node 131. Similarly, nodes 105 through 125 are connected to SAN 100 via ports 106, 111, 116, 121 and 126, respectively.

SAN 100 also includes network devices 135, 140 and 145. Such network devices may be of any kind known in the art, such as routers, switches, bridges, etc. These network devices are connected to their respective nodes by fabric ports. For example, network device 135 is connected to nodes 105 and 110 by fabric ports 150 and 155, respectively. Such ports are designated with an "F" in Fig. 1.

Connections between network devices are made by expansion ports or "E" ports. Connections between E ports are referred to as Inter-Switch Links ("ISLs"). For example, network device 135 is connected to network device 140 via an ISL between E port 160 of network device 135 and E port 170 of network device 140. Similarly, the connection between network device 140 and 145 is made by an ISL between E ports 175 and 180.

As is well known in the art, connections between network devices and nodes of storage area networks are commonly made via optical fiber. Data are transmitted on such networks according to various formats, but most commonly using the Fiber Channel protocol.

Some network devices may be configured to support a novel frame format, known as extended inter-switch link ("EISL") format, which is the subject of other pending patent applications assigned to Andiamo Systems. The description of some embodiments and applications of EISL in U.S. Patent Application Number 10/034,160 is hereby incorporated by reference for all purposes. In one example, the EISL format allows a single network device to process frames or packets having different formats. For example, a network device configured to support EISL may process both FC frames and Ethernet frames. The EISL format also supports VLANs, VSANs and similar features.

An EISL format allows the implementation of a fibre channel network with features and functionality beyond that provided by ISL format. In one example, the EISL format allows a port (known herein as a "trunking port") to transport frames of more than one format. For example, a trunking port can switch Ethernet and Fiber Channel ("FC") frames and is adaptable to transmitting frames of other formats as they are developed. An EISL header is used on EISL links to enable this transportation of different frame types. In another example, the EISL format allows the implementation of multiple virtual storage area networks (VSANs) on a single physical network. In still other examples, the EISL format provides mechanisms for implementing forwarding mechanisms such as Multi-Protocol Label Switching (MPLS) or Time To Live (TTL) fields specifying how packets should be forwarded and when packets or frames should be dropped. Any format allowing for the implementation of multiple virtual storage area networks on a physical fibre channel network while also allowing the transmission of different frame types, forwarding fields, and/or time to live, etc. is referred to herein as an EISL format.

Fig. 2 indicates one example of an EISL frame. One of skill in the art will appreciate that the size, sequence and functionality of the fields within this EISL frame can vary from implementation to implementation. For example, the numbers of bits indicated for each field are different in alternative EISL frames.

The EISL frame 200 is bounded by start of frame delimiter ("SOF") 205 an end of frame delimiter ("EOF") 280. These delimiters enable an EISL-capable port to receive frames in a standard format at all times. If an EISL-capable port is not in EISL mode and receives frames in the EISL format, it accepts the frame according to some aspects of the invention. However, the port may not be able to send frames in EISL format.

In this embodiment, EISL header 260 includes VSAN field 240, which specifies the virtual storage area network number of payload 270. A VSAN allows for multiple logical or "virtual" storage area networks to be based upon a single physical storage area network. Accordingly, VSAN field 240 of EISL header 260 indicates the virtual storage area network to which this frame belongs.

MPLS label stack field 265 provides a common forwarding mechanism for both FC and Ethernet frames. Cyclic redundancy check ("CRC") field 275 is used for error detection.

Exchange Link Parameter ("ELP") protocol is an existing FC protocol that is used for communication with E ports. Similarly, Exchange Switch Capability

("ESC") protocol is an existing FC protocol that is used for communication between E ports. These protocols can be used to exchange information regarding the capabilities of network devices.

According to some aspects of the invention, a new protocol, known herein as exchange peer protocol ("EPP"), is provided for communication between E ports. According to some preferred aspects of the invention, the EPP protocol is used after the ESC protocol. In such implementations, ESC protocol is used to determine if a network device is capable of performing EPP protocol exchange. The EPP protocol may be used, for example, to determine the port VSAN of a peer port of a network device or to determine whether the peer port is configurable for supporting EISL. When the peer port is enabled for EISL, the peer port is referred to as a "trunking port".

Fig. 3 illustrates a simplified version of an EISL frame. Here, frame 300 includes EISL header 305, header 310 and payload 315. Header 310 may be, for example, an FC header or an Ethernet header. According to some aspects of the present invention, field 320 is a field of payload 315. In one example, field 320 is a service access point ("SAP") field, which is a part of a fiber channel frame that is reserved for services that may be defined by a client. Field 320, according to some aspects of the invention, is an SAP field used for encoding EPP. According to some such aspects of the invention, field 320 is an EPP header and payload 315 includes an EPP payload, which will be described in more detail below.

Fig. 4 illustrates stack 400 according to some embodiments of the present invention. Stack 400 includes physical layer 405. For simplicity, all of the fiber channel layers are illustrated as a single layer, FC 2 layer 410. Switch Interlink Services ("SW_ILS") layer 415 provides functionality for ELP 420 and ESC 425, according to the standard FC format. Layer 415 also provides a mechanism for vendors to add their own protocols, such as EPP_ILS 430 in this example. The EPP protocol frames exchanged according to SW_ILS service specification are called EPP_ILS frames.

However, not all ports will recognize SW_ILS. Accordingly, in other implementations of the present invention, other formats or services may be used to provide EPP services. For example, other implementations of the invention use Extended Link Services (ELS) format to provide EPP services.

Fig. 5 is a flow diagram that depicts an exchange of information between two E ports according to some aspects of the present invention. E port A may be, for

example, port 160 of Fig. 1 and E port B may be, for example, E port 170 of Fig. 1. In other embodiments, one or both ports are F ports and may exchange frames using, for example, ELS format.

The information exchanged in section 505 of Fig. 5 represents the detection phase of EPP, wherein the EPP capability of an attached peer port is detected. Detection phase 505 is performed using ELP and ESC according to one implementation of this method.

Area 510 represents the SYNC phase of EPP, wherein configuration information of interest to the peer port is exchanged. According to some such embodiments, the configuration information is exchanged in time-length-value ("TLV") format, which will be described below with reference to Fig. 5A.

Finally, area 515 represents the commit phase of EPP. In the commit phase, the results of the exchange of configuration information that took place during the SYNC phase are applied to hardware and/or software of the peer ports, as needed. In the implementation illustrated in Fig. 5, the EPP detection phase 505 uses ESC service exchanges during E-port initialization. In ESC, the originator port can publish the protocol/services supported by the originator port. The peer port is required to respond with the service it agrees to work with or it can respond as "command unsupported."

At time 520, a link has been established between port A and port B. In step 525, port A sends an ELP request to port B. In this instance, port A has initiated the process. However, as will be explained in more detail below, the present invention includes a mechanism for dealing with situations in which both ports A and B have simultaneously initiated the process. ELP request 525 includes link-level parameters such as buffer-to-buffer credit (indicating how much data can be transmitted from one buffer to another before new credits are required).

In step 530, port B sends information to port A indicating an acceptance of the ELP request. In essence, step 525 involves the sending of port A's link-level parameters to port B and step 530 involves the sending of port B's link-level parameters to port A. In step 535, port A sends an acknowledgement to port B. At this time, port A knows port B's link configuration and port B knows port A's link configuration.

Then, in step 540, port A sends other information regarding the configuration of the network device that includes port A. In this step, port A indicates the services/protocols that port A can support. In some embodiments, the information will

include a vendor string that indicates the particular vendor and model number of the network device and its capabilities. In one such embodiment, step 540 includes the transmission of services/protocols that port A can support in code/service pairs. Some codes may be standard FC codes which correspond with standard FC services (e.g., FSPF). However, one such code is a unique code that corresponds with EPP. In step 545, port B sends an acceptance to port A and also sends information regarding the vendor and switch capabilities of the switch associated with port B. In this example, both port A and port B support EPP. Accordingly, detection phase 505 was successful and in steps 530 and 545, port B accepted port A's request and ESC information, respectively. However, port B could have rejected either of those requests. Alternatively, port B could have selected a different service if port B did not support EPP. .

The combination of a request and an acceptance (or of a request and a rejection) will sometimes be referred to herein as an "exchange." In the embodiment described with respect to Fig. 5, the exchanges are performed according to an SW_ILS format, as described above.

After determining that port B supports EPP and that port B could be configured to be a trunking port, port A sends an EPP_SYNC_ILS to port B in step 550 and EPP SYNC_ILS phase 510 begins. In this embodiment, the EPP_SYNC_ILS includes configuration information for use by Port B in configuring itself to be a trunking E port. However, in other embodiments, EPP may be used for port VSAN consistency checks without configuring port B as a trunking port.

Fig. 5A illustrates frame 585 in type-length-value ("TLV") format, which is a preferred format for data exchanged between ports A and B during SYNC phase 510. Type field 590 encodes how value field 592 is to be interpreted. In other words, type field 590 indicates what kind of value will be encoded in value field 592. Length field 591 indicates the length of value field 592, e.g., in bytes. Value field 592 is a payload that encodes information to be interpreted as specified by type field 590.

TLV format is inherently quite flexible, because both the type and length of value field 592 can be varied. However, in other embodiments of the invention, fixed-length frames may be used for the same purpose.

Referring again to Fig. 5, the exchange of trunking information will be described. As noted above, trunking information is one type of information that may be exchanged during step 550 of SYNC phase 510. According to some embodiments of the present invention, trunking configuration information includes admin trunk

mode information (administratively configured by the user), which may be "ON," "OFF" or "AUTO." "OFF" indicates that the sending port is configured not to operate as a trunking port. "ON" indicates that the sending port can operate as a trunking port if the receiving port does not explicitly prohibit this from happening. "AUTO" indicates that the sending port can operate as a trunking port if the receiving port is configured with trunking mode "ON."

Fig. 6 is table that indicates trunk mode negotiation according to some aspects of the present invention. If the sending trunk mode (here, port A) has an admin trunk mode setting of "OFF," then the sending port will be treated as a non-trunking port. If the admin trunk mode of the sending port is "ON," the sending port will be treated as a trunking port if the receiving port (here, port B) has an admin trunk mode of "ON" or "AUTO." If the sending port has an admin trunk mode of "AUTO," the receiving port must have an admin trunk mode of ON for the sending trunk mode to operate as a trunking port. Otherwise, the receiving port will operate as a normal port.

Referring again to Fig. 5, in step 555 port B sends an acknowledgement to port A. In step 560, port B sends its own configuration information, which may include trunking configuration information as described above, to port A.

In addition to exchanging admin trunk mode information, ports A and B may exchange VSAN list information during SYNC phase 510. The exchange of VSAN list information according to one such implementation will now be explained with reference to Fig. 7. In this example, ports A and B exchange bit maps that indicate which VSANs to allow. Here, port A sends bit map 705 to port B in which bits 1 through 5 have a value of "1," indicating that VSANs 1 through 5 should be allowed. Port B, in turn, sends bit map 710 indicating that VSANs 4 through 8 should be allowed. In preferred implementations, the bit maps indicate the status of more (or less) than 8 VSANs and include a correspondingly greater (or smaller) number of bits.

Both ports A and B, or the network devices associated with the respective ports, then compute an intersection bit map that indicates the VSANs common to both ports. In this case, intersection bit map 715 indicates that VSANs 4 and 5 are both allowed. In some embodiments of the present invention, the intersection bit map is computed at the end of the EPP_SYNC phase. In other embodiments of the present invention, the intersection bit map is computed at other times. However, this process should occur prior to the beginning of the commit phase.

After the intersection bit map has been computed, the network devices associated with ports A and B each will store the intersection bit map in memory and

only VSANs 4 and 5 will be permitted to send data frames along this data path. VSANs 4 and 5 are known as "operational VSANs" on the link between port A and port B.

According to some embodiments of the present invention, the configuration information exchanged during SYNC phase 510 includes port VSAN information. In some such aspects of the invention, port VSAN information is particularly important when the ports are functioning as non-trunking ports. If ports are functioning as trunking ports, the EISL header will contain a VSAN number indicating the VSAN to which the frame belongs.

However, according to some aspects of the invention, if the ports are not functioning as trunking ports, there will be no EISL header and consequently no VSAN number. If a port is not trunking, frames will be transmitted in the native FC format, not in EISL format. However, a VSAN will be implicitly associated with each frame. This VSAN is the port VSAN of the receiving port.

By default, every E port has a port VSAN number equal to 1. However, various port VSAN numbers may be assigned. If there is a mismatch between port VSAN numbers, various actions may take place according to various aspects of the present invention. According to some such aspects, a system administrator would be notified if, for example, a port having a port VSAN number of 1 sent a packet to a port having a port VSAN number of 2. According to other aspects of the invention, one or more of the ports would be brought down in the event of such a port VSAN mismatch.

At the end of step 560, port A knows the configuration of port B and port B knows the configuration of port A. In step 565, port A sends an acknowledgement to port B indicating that it has received port B's EPP_SYNC configuration information. Then, the EPP_SYNC phase of the process has concluded. On completion of SYNC phase 510, ports A and B will evaluate the configuration information that needs to be applied.

In the current example, ports A and B are configured to become trunking E ports. Accordingly, prior to EPP_Commit phase 515, port B is configured to be a trunking E port in programming step 568. According to some aspects of the invention, programming step 568 involves hardware programming necessary for supporting trunking mode operation and the preparation of EISL frames. In one instance, when the port is enabled for trunking mode, all frames are transmitted in EISL format.

When step 568 is complete, the EPP_Commit phase commences in step 570 by the sending of an EPP_Commit request from port B to port A. After port A receives

the EPP_Commit request, port A performs its own programming operation in step 572, which is parallel to the programming step 568 of port B: according to some aspects of the invention, programming step 572 involves hardware programming necessary for supporting trunking mode operation and the preparation of EISL frames. In one instance, when the port is enabled for trunking mode, all frames are transmitted in EISL format. Then, in step 575, port A sends an SW_ACC to port B, indicating that port A has completed its programming step.

Then, in step 580, port B sends an acknowledgement to port A indicating receipt of the SW_ACC sent in step 575 and completion of the EPP commit exchange on its side. At this time, port A has completed the EPP commit exchange. In the present example, this means that ports A and B are now configured for trunk mode operation

At some time after ports A and B have been transmitting data, an operator may decide to reconfigure some aspect of the ports. For example, the VSAN number may change on one or both of the ports and a new intersection bit map would need to be computed. If this is the case, the foregoing process need not go back through the ELP and ESC phases, but may proceed directly to the EPP_SYNC and EPP_Commit phases.

This process will be outlined with reference to Fig. 7A. In step 750, a network administrator changes the local admin trunk mode of port A from "AUTO" to "ON." In step 755, the EPP SYNC process begins with a parallel to step 550 of Fig. 5, in which the new local admin trunk mode of port A is transmitted to port B. In step 760, port B sends an "ACK" to port A. In this example, the peer admin trunk mode (of port B) remains set to "AUTO." Consequently, port B sends its peer admin trunk mode to port A in step 765, port A responds with an "ACK" in step 770 and both ports change their operational trunk mode to T (trunking) in step 775. The necessary EPP commit programming for trunking operation is performed in step 780.

Fig. 8 is a flow chart that depicts the process flow of an EPP method from the initiating port's perspective, according to one aspect of the present invention. The first step is step 805, the ready state. In step 810, an EPP_SYNC request is sent to the receiving port. In step 815, the initiating port requests an acceptance from the receiving port for the EPP_SYNC request. If the response is received within a predetermined time, the response is evaluated in step 820. If the response is not received within the predetermined time, the method proceeds to step 830 and the initiating port enters a retry waiting state.

Sometimes port B will send its own EPP_SYNC request during the time port A is awaiting a response to port A's EPP_SYNC request. This circumstance is known as a "collision." In the event of a collision, in step 816 port A determines whether to accept the EPP_SYNC request from port B. If port A does accept the EPP_SYNC request from port B, the process continues to step 910 of Fig. 9, which is described below. If port A does not accept the EPP_SYNC request from port B, port A sends a rejection (e.g., an "SW_RJT") to port B in step 817. Then, the process returns to step 815.

In step 835, it is determined whether the retry count or time is exceeded. If this retry count is exceeded, a failure will be reported and the system will return to a ready state. If the retry count is not exceeded, the EPP_SYNC request will be sent once again in step 810 and the process will proceed from step 810.

In step 820, if the response is determined to be acceptable, the method proceeds to step 825, where the system waits for an EPP_Commit state. If the response is determined not to be acceptable in step 820, an SW_RJT response is sent to the receiving port and the initiating port returns to the ready state of step 805.

If an EPP_Commit is received by the initiating port in step 825, then the process continues to step 840, wherein hardware programming is performed on the initiating port. In step 845, it is determined whether the hardware programming is completed. If not, the method proceeds to step 855, wherein the hardware programming step is reported and the system enters the retry condition of step 830. If the hardware programming is a success, the method proceeds to step 850 and an SW_ACC response for the EPP_Commit is transmitted to the receiving port.

The process then continues to step 860, wherein the initiating port waits for an acknowledgement from the receiving port. If the acknowledgement is not received within a predetermined time, then the process proceeds to step 855. If the acknowledgement is received within the predetermined time, the initiating port returns to the steady state of step 805.

Fig. 9 indicates the EPP process from the perspective of the receiving port. In step 905, the receiving port is in a ready state. In step 910, an SW_ACC is sent to the initiating port for the EPP_SYNC. In step 915, the receiving port waits for an acknowledgement for the SW_ACC response. If this response is not received within a predetermined time, there is a timeout and the receiving port returns to the ready state of step 905. If the acknowledgement is received within the predetermined, the method proceeds to step 920 and hardware programming is performed on the receiving port.

In 925 it is determined whether the hardware programming is completed. If not, a failure report is made in step 930 and the receiving port returns to a ready state in step 905. If the hardware programming is done, the method proceeds to step 935 and an EPP_Commit is sent to the initiating port.

In step 940, the receiving port waits for an SW_ACC for the EPP_Commit that it has sent to the initiating port. If no such response is received within a predetermined time, the process proceeds to step 930 and a failure is reported. The receiving port then returns to the ready state of step 905. If a response is received during the predetermined time, then the method proceeds to step 945 and the response is evaluated. If the response is determined to be acceptable, a success is notified. In step 950, if the response is not determined to be acceptable, an error is reported and the system returns to the ready state of 905.

Fig. 10 indicates the components, values and sizes of EPP header fields according to some embodiments of the present invention. Other embodiments may have more or fewer fields. Moreover, the fields may have lengths other than those depicted in Fig. 10.

In one implementation of the present invention that uses SW_ILS, the command identifier field indicates values chosen from a range of vendor specific command identifiers. The command identifier may indicate, for example, an EPP request, an SW_RJT (reject) or an SW_ACC (accept). In one embodiment, the value of the command ID is 0X71000000. The revision field identifies the revision of the EPP service. For the first revision, the value is 1. The revision number should be changed every time there is a change in the EPP header.

As noted above, in some implementations EPP uses a two-phase mechanism to establish the operating environment. The first phase is the synchronizing phase (EPP_SYNC), where the configuration information on both sides is synchronized. The second phase is the commit phase (EPP_COMMIT), where the actual hardware programming is performed. The EPP command code field is used to identify whether the EPP request sequence is from the EPP_SYNC phase or the EPP_COMMIT phase.

The session field is used to identify a particular session on the side that initiated the EPP request sequence. In some cases of error or failure, EPP will retry its protocol exchange. The session number will be changed for each retry of the EPP operation. This feature helps identify stale sessions.

The worldwide name (WWN) indicates the WWN of the network device to which the port belongs. According to some aspects of the present invention, the

WWN information is used for resolving "collisions" of simultaneous EPP_SYNC requests.

The payload length field is used to identify the total length of the payload, including the EPP header. The reserved field is reserved for future use.

There will be times when 2 ports will simultaneously send EPP requests to one another. Such "collisions" will be resolved based on the WWN of the network device with which the port is associated. The port within the network device having the lower WWN will send an SW_ACC to the other port. The port whose network device has the WWN will send SW_RJT to the other port, with a reason code indicating collision.

Generally, the techniques of the present invention may be implemented on software and/or hardware. For example, they can be implemented in an operating system kernel, in a separate user process, in a library package bound into network applications, on a specially constructed machine, or on a network interface card. In a specific embodiment of this invention, the technique of the present invention is implemented in software such as an operating system or in an application running on an operating system.

A software or software/hardware hybrid implementation of the techniques of this invention may be implemented on a general-purpose programmable machine selectively activated or reconfigured by a computer program stored in memory. Such a programmable machine may be a network device designed to handle network traffic, such as, for example, a router or a switch. Such network devices may have multiple network interfaces including frame relay and ISDN interfaces, for example. Specific examples of such network devices include routers and switches.

For example, the methods of this invention may be implemented in specially configured network devices such as the MDS 9000 family of switches manufactured by Cisco Systems, Inc. of San Jose, California. A generalized architecture for some such machines will appear from the description given below. In an alternative embodiment, the techniques of this invention may be implemented on a general-purpose network host machine such as a personal computer or workstation. Further, the invention may be at least partially implemented on a card (e.g., an interface card) for a network device or a general-purpose computing device.

Referring now to Fig. 11, a network device 1160 suitable for implementing the techniques of the present invention includes a master central processing unit (CPU) 1162, interfaces 1168, and a bus 1167 (e.g., a PCI bus). When acting under the

control of appropriate software or firmware, the CPU 1162 may be responsible for implementing specific functions associated with the functions of a desired network device. For example, when configured as an intermediate router, the CPU 1162 may be responsible for analyzing packets, encapsulating packets, and forwarding packets for transmission to a set-top box. The CPU 1162 preferably accomplishes all these functions under the control of software including an operating system (e.g. Windows NT), and any appropriate applications software.

CPU 1162 may include one or more processors 1163 such as a processor from the Motorola family of microprocessors or the MIPS family of microprocessors. In an alternative embodiment, processor 1163 is specially designed hardware for controlling the operations of network device 1160. In a specific embodiment, a memory 1161 (such as non-volatile RAM and/or ROM) also forms part of CPU 1162. However, there are many different ways in which memory could be coupled to the system. Memory block 1161 may be used for a variety of purposes such as, for example, caching and/or storing data, programming instructions, etc.

The interfaces 1168 are typically provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receiving of data packets over the network and sometimes support other peripherals used with the network device 1160. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various very high-speed interfaces may be provided, such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces, ASI interfaces, DHEI interfaces and the like. Generally, these interfaces may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile RAM. The independent processors may control such communications intensive tasks as packet switching, media control and management. By providing separate processors for the communications intensive tasks, these interfaces allow the master microprocessor 1162 to efficiently perform routing computations, network diagnostics, security functions, etc.

Although the system shown in Fig. 11 illustrates one specific network device of the present invention, it is by no means the only network device architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc.

is often used. Further, other types of interfaces and media could also be used with the network device.

Regardless of the network device's configuration, it may employ one or more memories or memory modules (such as, for example, memory block 1165) configured to store data, program instructions for the general-purpose network operations and/or other information relating to the functionality of the techniques described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example.

Because such information and program instructions may be employed to implement the systems/methods described herein, the present invention relates to machine-readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM) and random access memory (RAM). The invention may also be embodied in a carrier wave traveling over an appropriate medium such as airwaves, optical lines, electric lines, etc. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

While the invention has been particularly shown and described with reference to specific embodiments thereof, it will be understood by those skilled in the art that changes in the form and details of the disclosed embodiments may be made without departing from the spirit or scope of the invention. For instance, it will be appreciated that at least a portion of the functions described herein that are performed by a network device such as a router, a switch and/or selected components thereof, may be implemented in another device. For example, these functions can be performed by a host device (e.g., a personal computer or workstation). Such a host can be operated, for example, by a network administrator. Considering these and other variations, the scope of the invention should be determined with reference to the appended claims.

WE CLAIM:

1. A method for modifying configurations of peer ports interconnecting network devices, the method comprising:

determining that the interconnected peer ports, comprising a first port of a first network device and a second port of a second network device, can support Exchange Peer Parameters protocol;

exchanging configuration information using the Exchange Peer Parameters protocol between the interconnected peer ports; and

configuring the interconnected peer ports according to the exchanged information.

2. The method of claim 1, wherein the configuration information comprises virtual storage area network information.

3. The method of claim 1, wherein the configuration information comprises trunk mode information.

4. The method of claim 1, wherein the configuration step further comprises configuring hardware of the interconnected peer ports according to the exchanged information.

5. The method of claim 1, wherein the configuration step further comprises configuring software of the interconnected peer ports according to the exchanged information.

6. A method for modifying a configuration of a network device, the method comprising:

determining that a first expansion port of a first network device, the first expansion port attached to a second expansion port of a second network device, can be configured to transmit frames in Extended Interswitch Link format; and

configuring the first expansion port to transmit frames in Extended Interswitch Link format.

7. The method of claim 6, wherein the determining step comprises exchanging trunk mode information between the first expansion port and the second expansion port via Exchange Peer Parameters protocol.

8. The method of claim 6, wherein the configuring step comprises configuring the hardware and/or software of the first expansion port to enable transmission of frames in Extended Interswitch Link format.

9. The method of claim 6, wherein the configuring step comprises informing the second expansion port via Exchange Peer Parameters protocol that the configurations have been applied to the first expansion port.

10. A computer program for causing a first expansion port of a first network device to modify a configuration of a second expansion port of a second network device, the computer program causing the first expansion port to perform the following steps:

determining that the second expansion port can be configured as a trunking port for transmitting frames in Extended Interswitch Link format; and

configuring the second expansion port as a trunking port.

11. The computer program of claim 10, wherein the determining step comprises exchanging information between the first expansion port and the second expansion port via Exchange Link Parameter protocol.

12. The computer program of claim 10, wherein the determining step comprises exchanging information between the first expansion port and the second expansion port via Exchange Switch Capability protocol.

13. The computer program of claim 10, wherein the configuring step comprises exchanging information between the first expansion port and the second expansion port via Exchange Peer Protocol.

14. An apparatus for modifying a configuration of a network device, the apparatus comprising:

means for determining that the interconnected peer ports, comprising a first port of a first network device and a second port of a second network device, can support Exchange Peer Parameters protocol;

means for exchanging configuration information using the Exchange Peer Parameters protocol between the interconnected peer ports; and

means for configuring the interconnected peer ports according to the exchanged information.

15. A first network device for modifying a configuration of a second network device, the first network device configured to perform the following steps:

determining that a port of the second network device can support Exchange Peer Parameter protocol; and

causing the port to be configured based on configuration information exchanged between the first network device and the port via Exchange Peer Parameters protocol.

16. The first network device of claim 15, wherein the determining step comprises exchanging information between the first network device and the port via Exchange Link Parameter protocol.

17. The first network device of claim 15, wherein the determining step comprises exchanging information between the first network device and the port via Exchange Switch Capability protocol.

18. The first network device of claim 15, wherein the configuring step comprises exchanging information between the first network device and the port via Exchange Peer Parameter protocol.

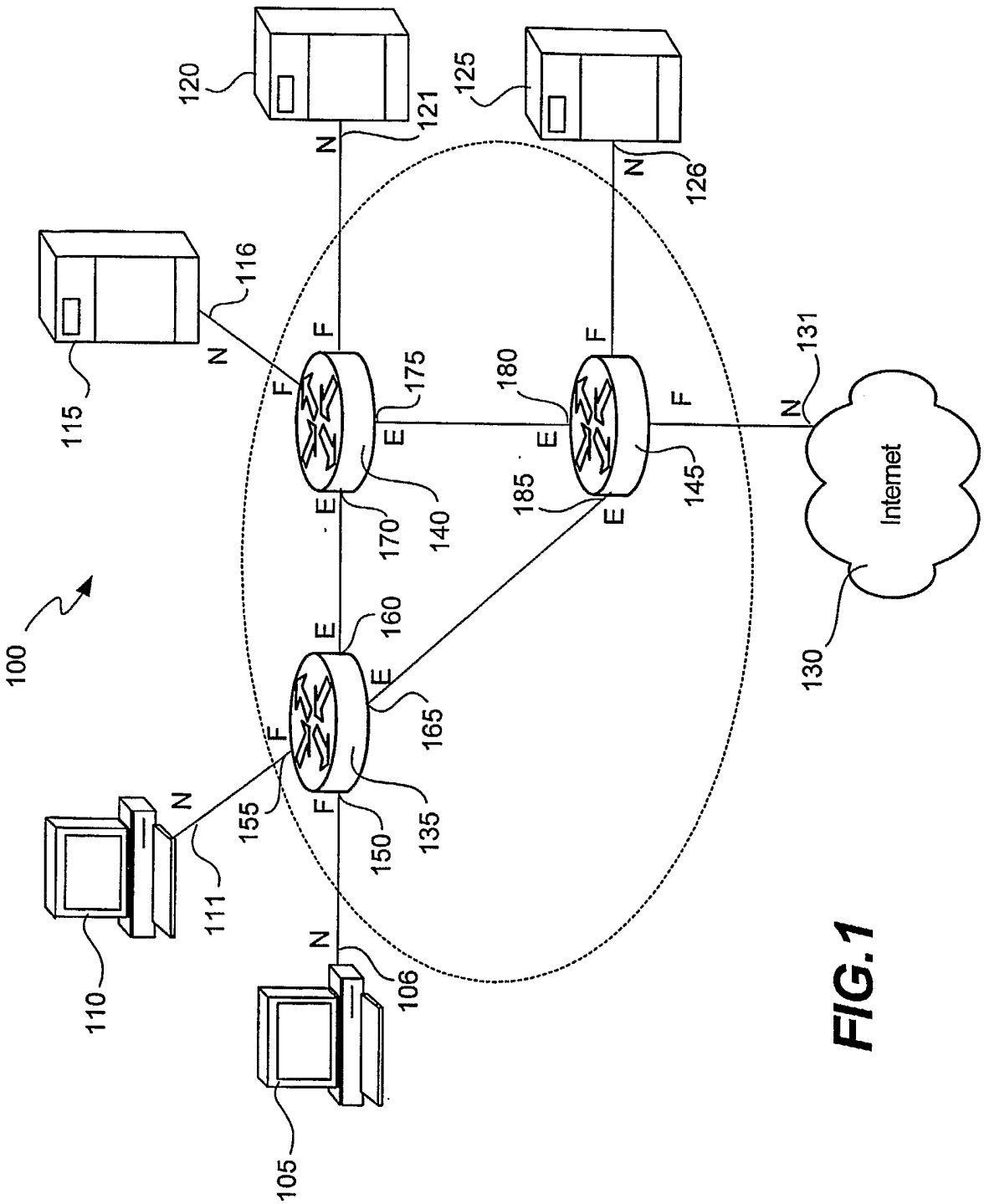

100
105
106
110
111
115
116
120
121
125
126
130
131
135
140
145
150
155
160
165
170
175
180
185
Internet
N
F
E

FIG.1

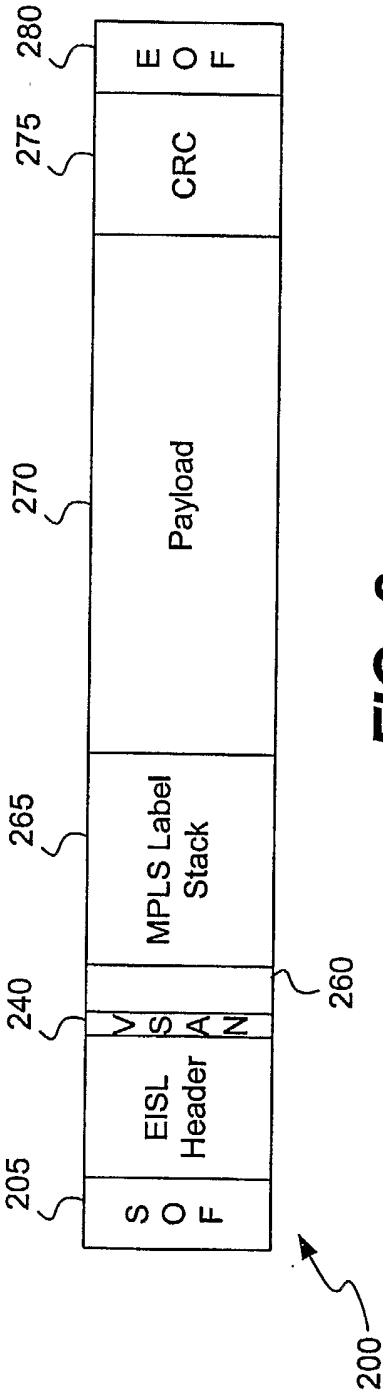

280
EOF
275
CRC
270
Payload
265
MPLS Label Stack
260
240
VSAN
EISL Header
205
SOF
200

FIG. 2

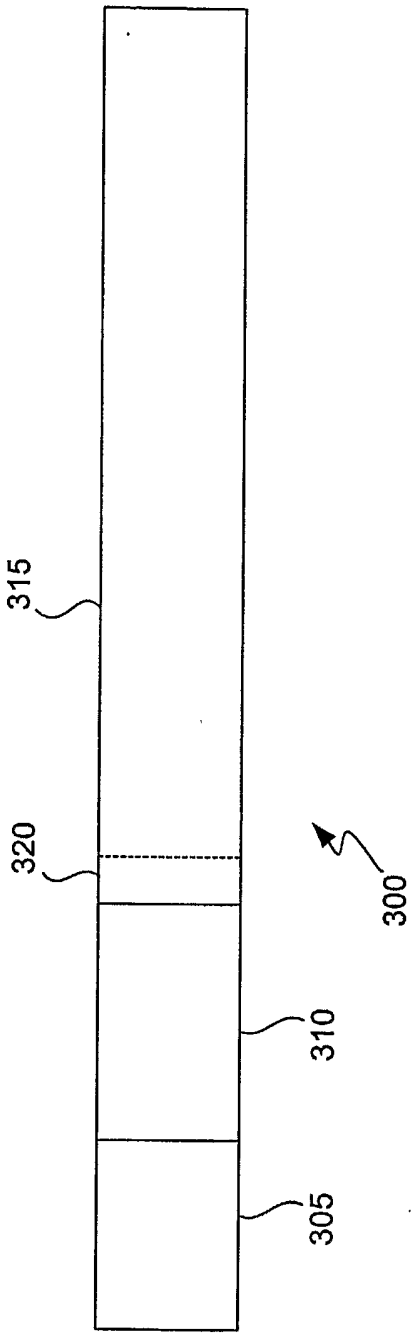


FIG. 3

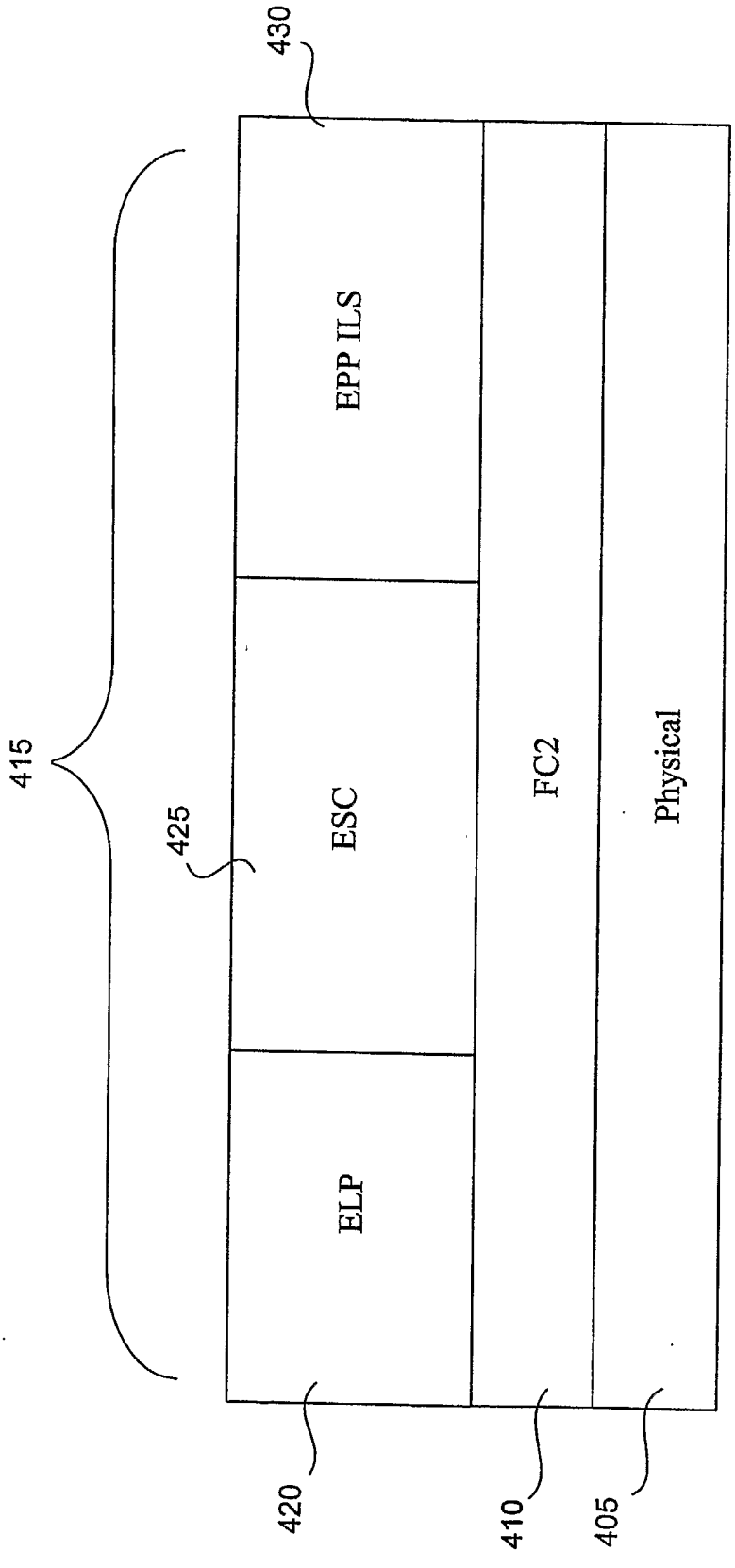


FIG. 4

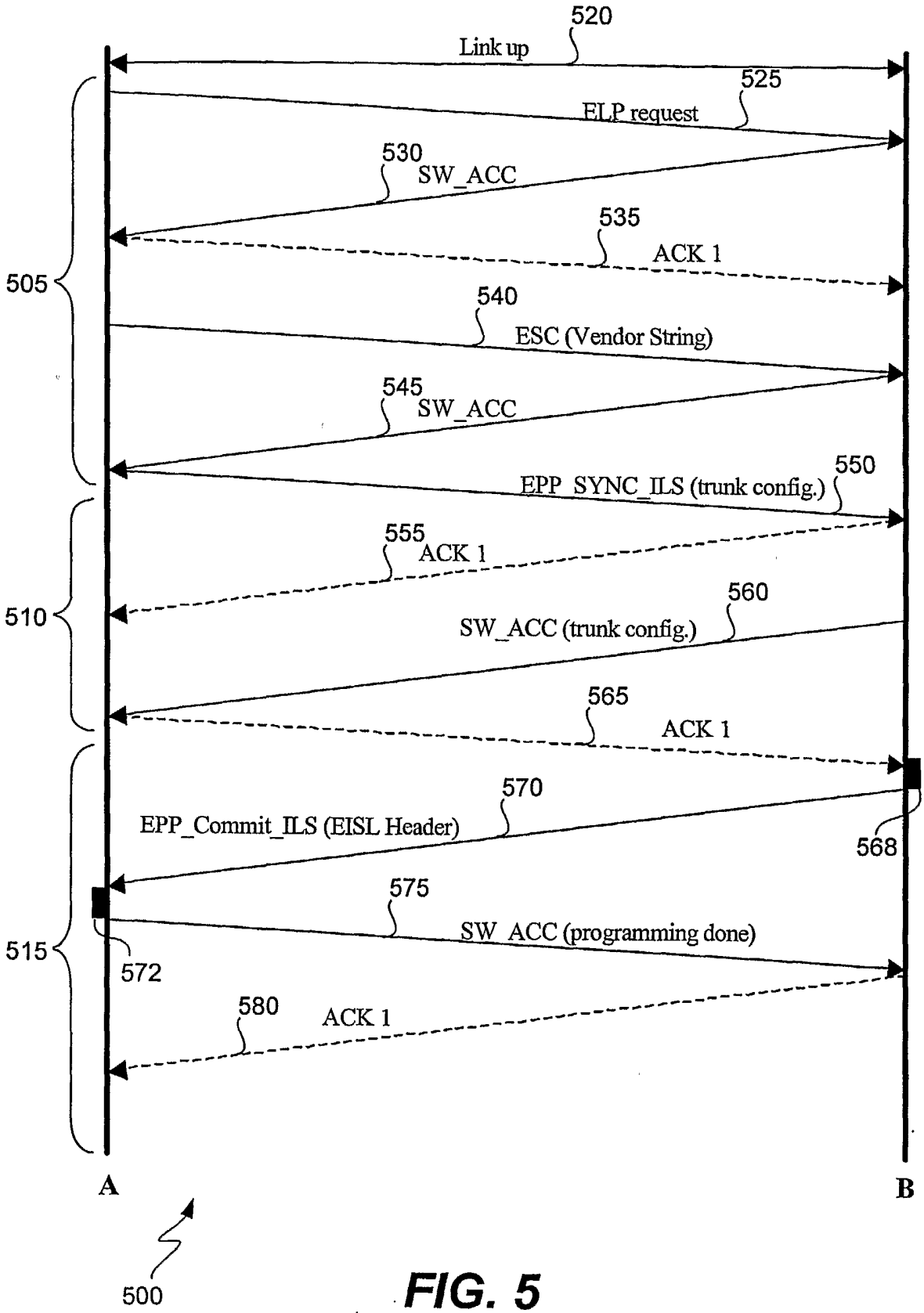

520
Link up
525
ELP request
530
SW_ACC
535
ACK 1
505
540
ESC (Vendor String)
545
SW_ACC
550
EPP_SYNC_ILS (trunk config.)
555
ACK 1
510
560
SW_ACC (trunk config.)
565
ACK 1
570
EPP_Commit_ILS (EISL Header)
568
575
SW_ACC (programming done)
515
572
580
ACK 1
A
B
500

FIG. 5

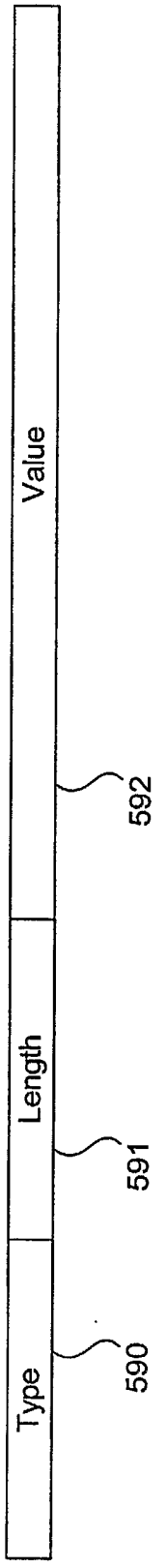


FIG. 5A

OPERATIONAL TRUNK MODE

	PEER ADMIN TRUNK MODE		
LOCAL ADMIN TRUNK MODE	OFF	ON	AUTO
OFF	NT	NT	NT
ON	NT	T	T
AUTO	NT	T	NT

NT=Non-trunking
T=trunking

FIG. 6

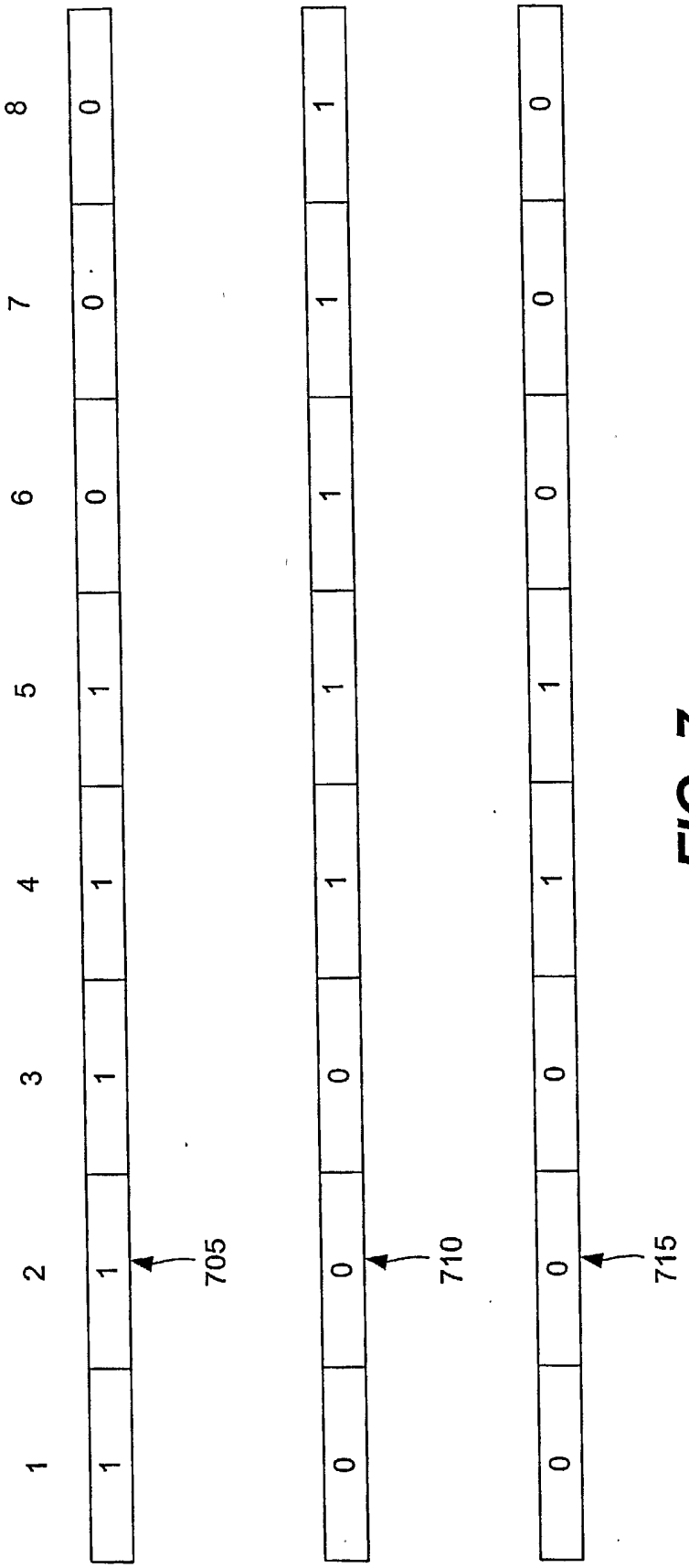


FIG. 7

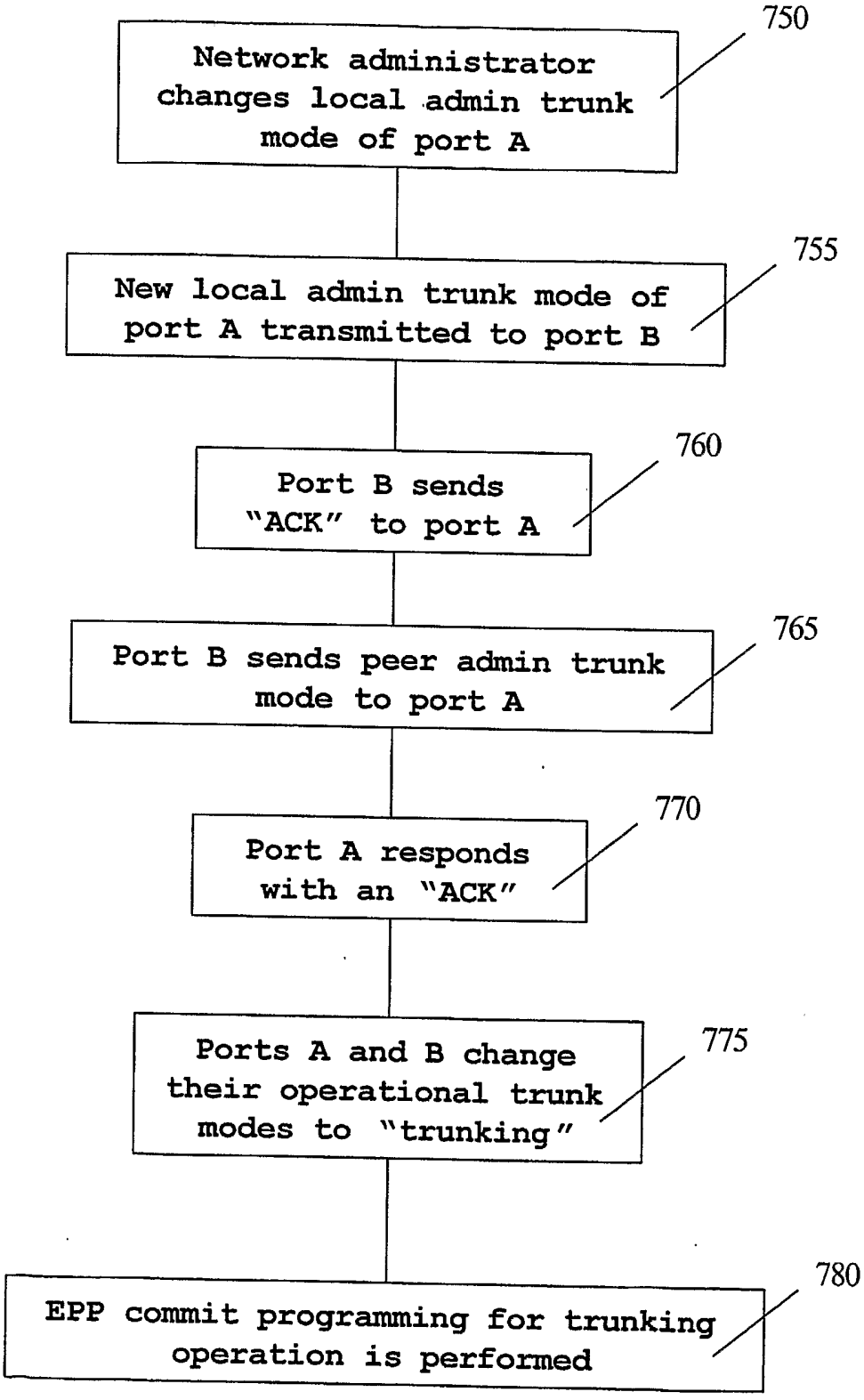

Network administrator changes local admin trunk mode of port A
750
New local admin trunk mode of port A transmitted to port B
755
Port B sends "ACK" to port A
760
Port B sends peer admin trunk mode to port A
765
Port A responds with an "ACK"
770
Ports A and B change their operational trunk modes to "trunking"
775
EPP commit programming for trunking operation is performed
780

FIG. 7A

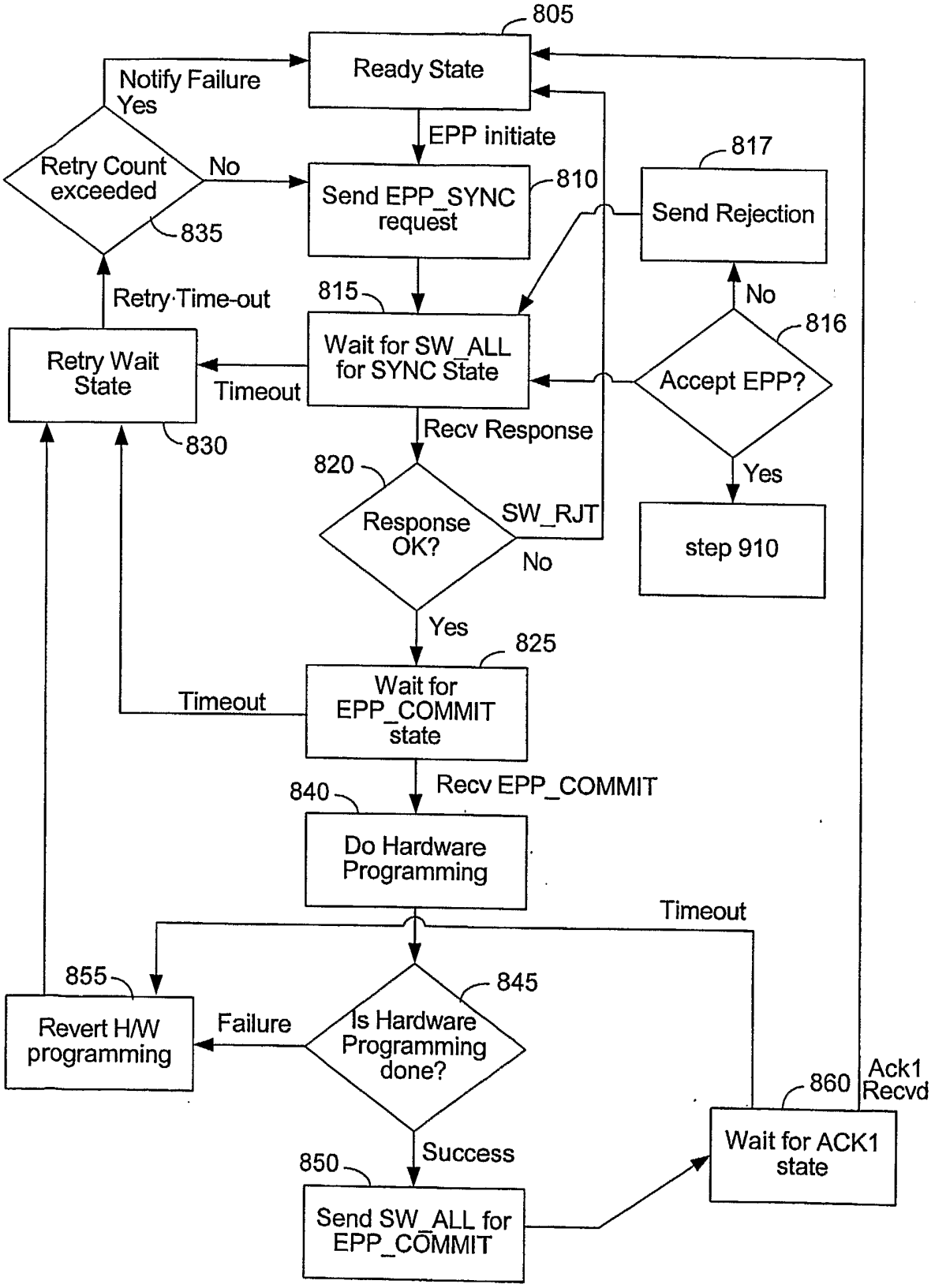

805
Ready State
Notify Failure
Yes
EPP initiate
Retry Count exceeded
No
835
Send EPP_SYNC request
810
817
Send Rejection
Retry Time-out
815
Wait for SW_ALL for SYNC State
No
816
Accept EPP?
Retry Wait State
Timeout
830
Recv Response
820
Response OK?
SW_RJT
No
Yes
step 910
Yes
825
Wait for EPP_COMMIT state
Timeout
Recv EPP_COMMIT
840
Do Hardware Programming
Timeout
855
Revert H/W programming
Failure
845
Is Hardware Programming done?
860
Ack1 Recvd
Wait for ACK1 state
Success
850
Send SW_ALL for EPP_COMMIT

FIG. 8

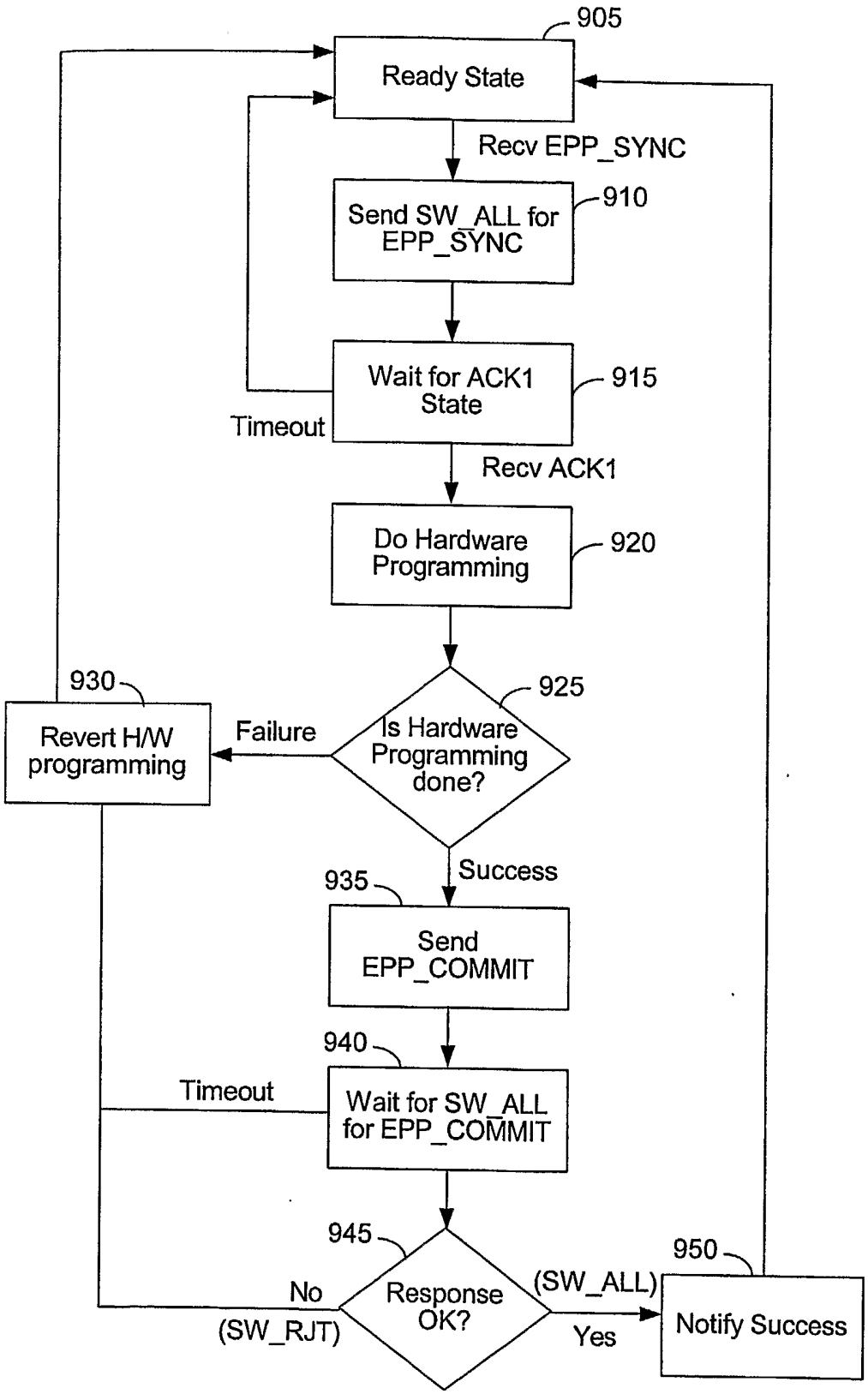

905
Ready State
Recv EPP_SYNC
Send SW_ALL for EPP_SYNC
910
Wait for ACK1 State
915
Timeout
Recv ACK1
Do Hardware Programming
920
930
Revert H/W programming
Failure
Is Hardware Programming done?
925
Success
935
Send EPP_COMMIT
940
Timeout
Wait for SW_ALL for EPP_COMMIT
945
950
No
(SW_RJT)
Response OK?
(SW_ALL)
Yes
Notify Success

FIG. 9

Item	Value	Size (bytes)
Command Id	0x71000000	4
Revision	1 (for first version)	1
EPP Command Code	0x0001 (EPP_SYNC) 0x0002 (EPP_COMMIT)	1
Session	0x0001 thru 0xffff	2
Switch WWN	Global switch Name	8
Reserved	0x0000	2
Payload Length	length of EPP payload (without FC header)	2

FIG. 10

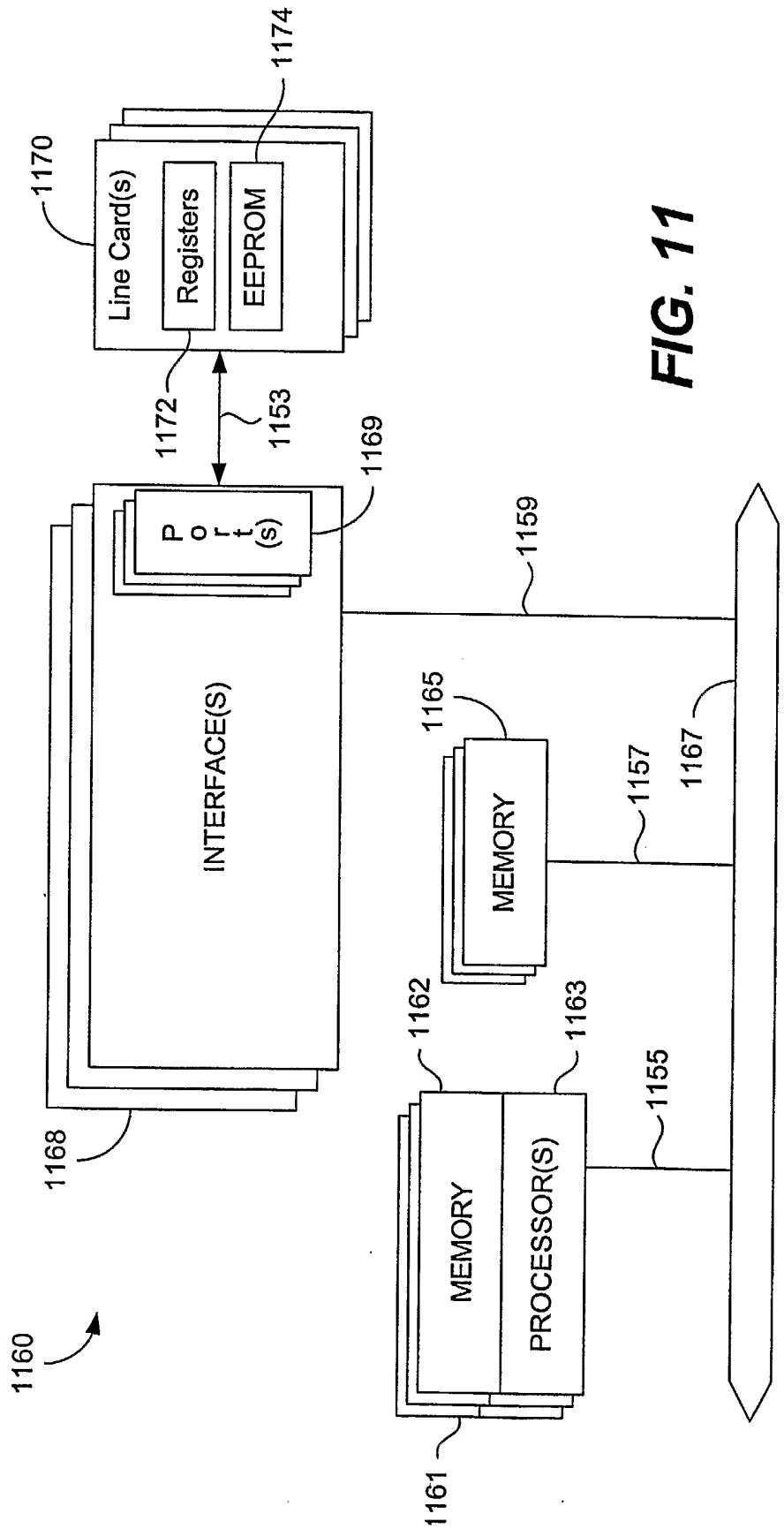

1174
1170
Line Card(s)
Registers
EEPROM
1172
1153
1169
Port(s)
1159
INTERFACE(S)
1165
MEMORY
1157
1167
1162
1163
1155
MEMORY
PROCESSOR(S)
1168
1160
1161

FIG. 11