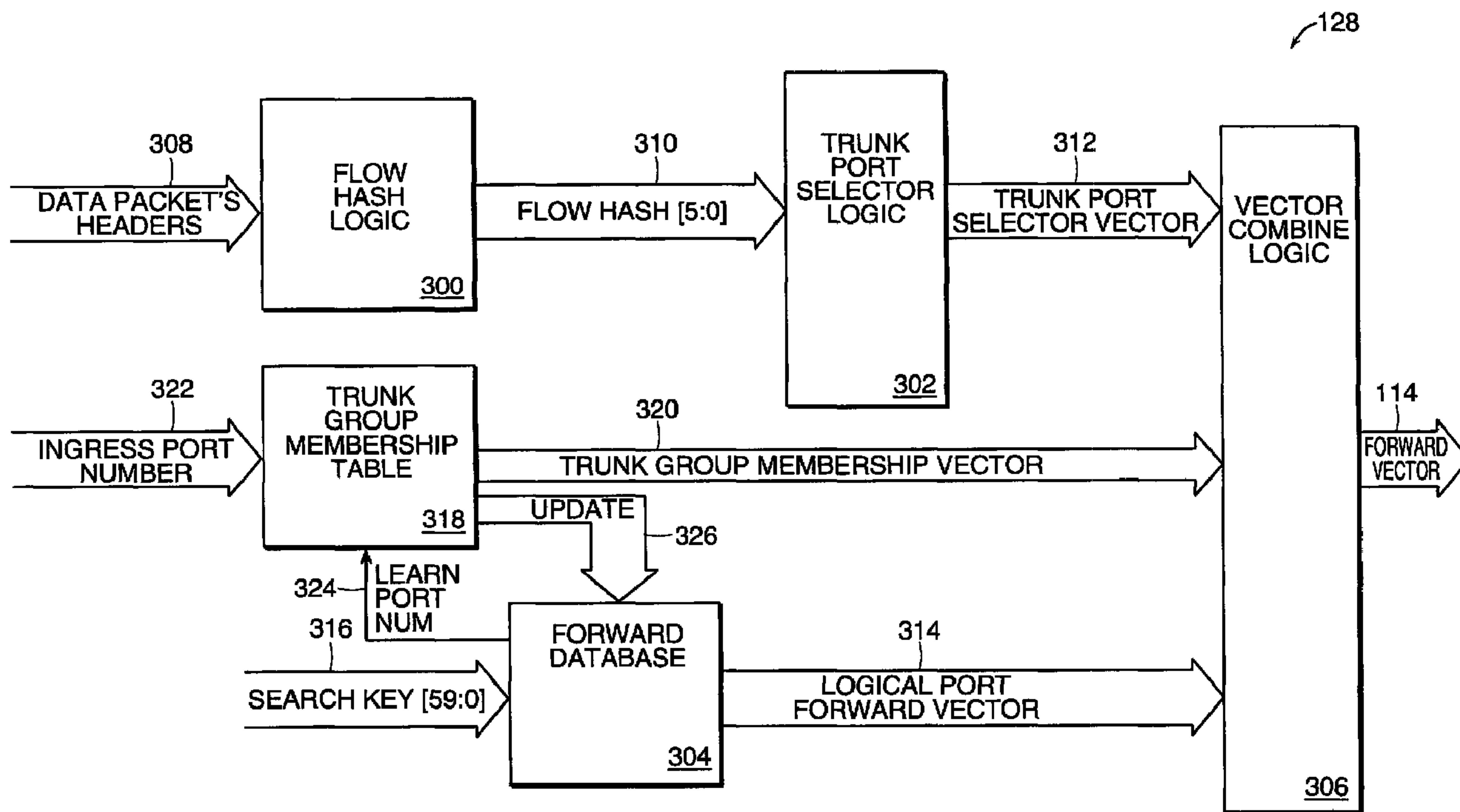




(86) Date de dépôt PCT/PCT Filing Date: 2001/01/17
 (87) Date publication PCT/PCT Publication Date: 2001/09/07
 (45) Date de délivrance/Issue Date: 2010/07/13
 (85) Entrée phase nationale/National Entry: 2002/08/22
 (86) N° demande PCT/PCT Application No.: CA 2001/000037
 (87) N° publication PCT/PCT Publication No.: 2001/065777
 (30) Priorité/Priority: 2000/02/29 (US09/516,001)

(51) Cl.Int./Int.Cl. *H04L 12/44* (2006.01),
H04L 12/56 (2006.01)
 (72) Inventeur/Inventor:
WYATT, RICHARD M., CA
 (73) Propriétaire/Owner:
MOSAID TECHNOLOGIES INCORPORATED, CA
 (74) Agent: BORDEN LADNER GERVAIS LLP

(54) Titre : AGREGATION DE LIGNES
 (54) Title: LINK AGGREGATION



(57) Abrégé/Abstract:

In a switch with multiple physical links to a destination, data is forwarded to the destination by distributing received data across the physical links. A flow hash is selected for the received data's data flow dependent on a destination address and source address included in the received data. The flow hash selects one of the physical links to the destination for a data flow but potentially a different physical link for a different data flow, thereby forwarding the received data by distributing the received data across the physical links while maintaining frame ordering within a data flow.

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

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
7 September 2001 (07.09.2001)

PCT

(10) International Publication Number
WO 01/65777 A1

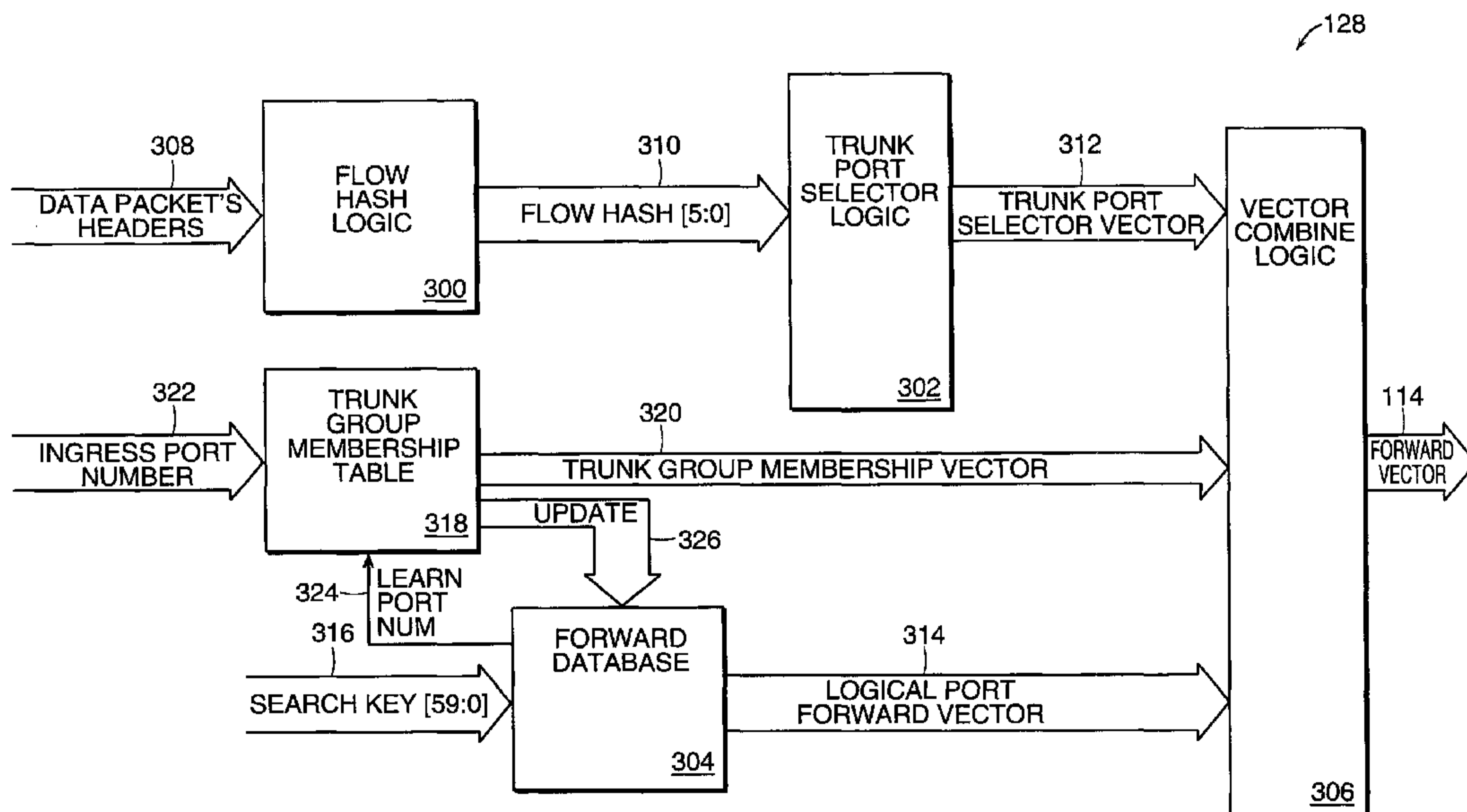
- (51) International Patent Classification⁷: **H04L 12/44, 12/56**
- (21) International Application Number: PCT/CA01/00037
- (22) International Filing Date: 17 January 2001 (17.01.2001)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
09/516,001 29 February 2000 (29.02.2000) US
- (71) Applicant (for all designated States except US): **MOSAID TECHNOLOGIES INCORPORATED** [CA/CA]; 11 Hines Road, Kanata, Ontario K2K 2X1 (CA).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): **WYATT, Richard, M.** [CA/CA]; 2148 Plessier Street, Ottawa, Ontario K1G 2X1 (CA).
- (74) Agent: **PILLAY, Kevin**; Fasken Martineau DuMoulin LLP, Toronto Dominion Bank Tower, Suite 4200, P.O. Box 20, Toronto-Dominion Centre, Toronto, Ontario M5K 1N6 (CA).
- (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, 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, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— with international search 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: LINK AGGREGATION



(57) **Abstract:** In a switch with multiple physical links to a destination, data is forwarded to the destination by distributing received data across the physical links. A flow hash is selected for the received data's data flow dependent on a destination address and source address included in the received data. The flow hash selects one of the physical links to the destination for a data flow but potentially a different physical link for a different data flow, thereby forwarding the received data by distributing the received data across the physical links while maintaining frame ordering within a data flow.

WO 01/65777 A1

-1-

LINK AGGREGATION

BACKGROUND OF THE INVENTION

A networking switch receives data packets from a number of ingress ports connected to the switch and provides the data packets to a number of egress ports connected to the switch. The switch determines the egress port to which the data packets are provided dependent on the destination address included in the data packet.

Typically, a destination is connected through one physical link to one egress port in the switch. A data packet received at an ingress port for the destination is forwarded through the switch to the egress port. The destination may be a computer, another switch or a router.

To increase the bandwidth to a destination; that is, the number of data packets that can be forwarded through the switch to a destination, the destination may be connected to more than one egress port through multiple physical links with each physical link terminating at an egress port. The multiple physical links are members of a logical link between the switch and the destination.

Providing multiple physical links to a destination is called link aggregation or trunking. Link aggregation for IEEE 802.3 is described in tutorials published by the IEEE 802.3ad group at <http://grouper.ieee.org/groups/802/3/trunk-study/tutorial>.

A data packet arriving at an ingress port in the switch may be forwarded through the switch on any one of the physical links in the logical link to the destination. Thus, link bandwidth is increased because data packets for a destination are distributed amongst the physical links. To achieve maximum bandwidth utilization on the logical link, data packets to the destination must be evenly distributed amongst the physical links to the destination.

However, when distributing received data packets amongst the physical links, data packets for a data flow cannot be mis-ordered through the switch.

-2-

SUMMARY OF THE INVENTION

A switch includes a logical link connecting a destination to the switch. The logical link includes physical links. The system assumes that flow hash logic in the switch indexes a flow hash dependent on a data flow encoded in the received data. Trunk port selector logic in the switch selects a trunk port entry dependent on the flow hash. The trunk port entry selects the physical link on which to forward the received data to the destination.

The data flow is encoded in the destination and source addresses stored in a header in the received data. The source and destination addresses may be Ethernet source and destination addresses, IP source and destination addresses, UDP source and destination port addresses or TCP source and destination port addresses.

The switch includes vector combine logic which selects a port in the switch corresponding to the physical link. The port is selected dependent on a combination of a logical port forward vector and the trunk port entry.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1A illustrates a logical link connecting a destination to a switch according to the principles of the present invention;

Fig. 1B illustrates a switch shown in Fig. 1A including forwarding logic for forwarding data packets received at an ingress port on one of a plurality of links in the logical link connecting the destination to the switch;

Fig. 2A illustrates a prior art data packet which may be received on an ingress port connected to a switch;

Fig. 2B illustrates a prior art Ethernet Data link layer (L2) header which may be included in the data packet shown in Fig. 2A;

Fig. 2C illustrates a prior art Internet Protocol (Network layer (L3)) header which may be included in the data packet shown in Fig. 2A;

Fig. 3 illustrates the forwarding logic shown in Fig. 1B;

Fig. 4 is a flow diagram of the functions performed in the flow hash logic shown in Fig. 3;

Fig. 5 illustrates the trunk port selector table shown in Fig. 3;

Fig. 6 illustrates the combination of one of the trunk port selector entries shown in Fig. 5, a group membership table vector entry and a logical port forward vector entry;

Fig. 7 is a flow diagram of the steps for using the contents of a trunk group membership vector to update a logical port forward vector stored in the forward database 304.

DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

Fig. 1A illustrates a logical link 134 connecting a destination 112c to a switch 100 according to the principles of the present invention. The logical link or trunk group 134 includes physical links 132c-e. Destination 112c is connected to the switch through physical links 132c-e. A received data packet for destination 112c may be forwarded on any one of the three physical links 132c-e to destination 112c. The switch 100 includes an egress port queue 130a-c corresponding to each physical link 132c-e in the logical link 134. The switch forwards data packets received from a source 102a to one of the egress port queues 130a-c.

The egress port queue 130a-c to which a received data packet is stored before being forwarded on the corresponding physical link 132c-e is dependent on the data flow, that is; the source address and destination address included in the data packet. By selecting a physical link dependent on a source address and destination address, data packets for the same data flow are always forwarded on the same physical link and thus are not mis-ordered in the switch.

For example, data packets 140a-c to be forwarded to destination 112c are received by the switch from source 102a. Each data packet 140a-c includes the source address for source 102a and the destination address for destination 112c. The

-4-

switch determines the data flow from the source and destination addresses stored in the data packets 132c-e. As each of the data packets 140a-c is received and stored in memory in the switch and the address of the data packet in memory is stored in the order that it is received in egress port queue 130a. Each of data packets 140a-c is forwarded on physical link 132c to destination 112c. Thus, data packets 140a-c for the data flow from source 102a to destination 112c are transmitted to destination 112 in the order that they are received by the switch 100.

Fig. 1B illustrates the switch 100 shown in Fig. 1A including forwarding logic 128 for forwarding data packets received at an ingress port on one of a plurality of physical links 132c-e. The switch 100 includes an ingress ports engine 104, a packet storage manger 106, a segment buffer memory 108 and an egress ports engine 110. The physical links 132c-e are members of a logical link 134 connecting destination 112c to the switch 100. Physical links 132f-g are members of logical link 140 connecting destination 112d to the switch 100. A single physical link 132a connects destination 112a to the switch 100 and a single physical link 132b connects destination 112b to the switch 100. Thus, if all physical links are the same speed, logical link 140 provides double the bandwidth to destination 112d as the single physical link 132a to destination 112a and logical link 134 provides three times the bandwidth destination 112a as single physical link 132b to destination 112b.

The switch 100 may include any combination of single physical links and logical links to a destination 112. A logical link may include any number of physical links. The physical links in a logical link may connect non-sequential ports to a destination 112, for example, logical link 134 connects non-sequential egress ports (egress port 2 136c, egress port 3 136d, and egress port 5 136f) to destination 112c. Alternatively, a logical link may connect consecutive ports to a destination, for example, logical link 140 connects consecutive egress ports (egress port 6 136g, egress port 7 136h) to destination 112d.

Thus, all egress ports 136a-h may be members of the same logical link, each egress port 136a-h may be a single physical link or the egress ports 136a-h may be configured in a combination of logical links and single physical links to destinations 112a-d.

-5-

The members of a logical link are not limited to physical links 132a-h of the same speed. For example, a 1 Gigabit Ethernet egress port may be a member of the same logical links as 100 Mbits Ethernet egress port.

A data packet received at an ingress port 138a-c from a source 102a-c is forwarded to one or more egress ports 136a-h dependent on the forward vector 114 generated by the forwarding logic 128 in the ingress ports engine 104. The forward vector 114 is dependent on a logical port forward vector stored in a forward database implemented in the forwarding logic 128.

The packet storage manager 106 stores the ingress data 116 received in the data packet in the segment buffer memory 108. The packet storage manager 106 also stores the address of the received ingress data 116 in the segment buffer memory 108 in one or more egress port queues 130 dependent on the state of the forward vector 114. The packet storage manager 106 is described in U.S. Patent No. 6,711,170, issued March 23, 2004.

The egress ports engine 110 through a select signal 120 selects an egress port queue 130 from which to forward the address of received ingress data 116 on address 122 to the segment buffer memory 108. The ingress data 116 stored in segment buffer memory 108 is forwarded on egress data 118 to an egress port 136 a-h. The egress port 136a-h to which the egress data 118 is forwarded is dependent on the forward vector 114.

The forward vector 114 selects an egress port queue 130 in which to store the address in segment buffer memory 108 at which the data packet is stored. The egress ports engine 110 through the select signal 120 selects an egress port queue 130. The address 122 is forwarded to segment buffer memory 108. The egress data 118 stored at the address 122 is forwarded to the egress port engine 110 and from the egress port engine 110 to an egress port 136a-h dependent on the selected egress port queue 130.

Destination 112c is connected to three egress ports (port 2, port 3, port 5) 136c-e through physical links 132c-e. The physical links 132c-e are members of a

-6-

logical link or trunk group 134. The members of the logical link 134 are not limited to the three egress ports 136c-e shown. The members of the logical link 134 may include any combination of egress ports 136a-h in the switch 100. The forward vector 114 includes a bit for each egress port 136a-h through which a received data packet may be forwarded. A received data packet for destination 112c may be forwarded on any one of the three physical links 132c-e to destination 112c. The forwarding logic 128 selects one of the three physical links 132c-e to destination 112c so that a data packet for a data flow from one source to a destination is always forwarded on the same physical link 132c-e to the destination 112c. For example, physical link 132e may be selected for forwarding all data packets received from source 102a to destination 112c.

Fig. 2A illustrates a prior art data packet 200 which may be received at an ingress port 136a-c (Fig. 1B) connected to the switch 100 (Fig. 1B). Fig. 2B illustrates a prior art Ethernet header which may be included in the data packet 200 shown in Fig. 2A. Fig. 2C illustrates a prior art Internet Protocol (“IP”) header which may be included in the data packet 200 shown in Fig. 2A.

Fig. 2A shows a prior art data packet 200. The data packet 200 includes a data payload 210 and headers for networking layers 202, 204, 206, 208. Headers for four of the layers in the OSI model are shown, the physical layer (L1) header 202, the data link layer (L2) header 204, the networking layer (L3) header 206 and the transport layer (L4) header 208. For example, the data link layer (L2) header 204 may be Ethernet and the networking layer (L3) header 206 may be IP. The data packet 200 also includes a checksum 212.

Fig. 2B illustrates the format of a prior art Ethernet data link (L2) header 204. The Ethernet data link (L2) header 204 includes a device address for the destination node 104 (Fig. 1B); that is, the L2 destination address 214, and a device address for the source node 102 (Fig. 1B); that is, the L2 source address 216, an optional Virtual Local Area Network Identification (“VLAN ID”) field 218 and a length/type field 220. The VLAN ID 218 includes a Tag Protocol Identifier (“TPI”) field 218a and a Tag Control Information (“TCI”) field 218b. The VLAN ID field 218 provides support for VLAN switching based on IEEE 802.1Q tagging and IEEE 802.ID 1988 (802.1p) priority levels.

FIG. 2C illustrates the format of a prior art IP network layer (L3) header 206. The IP network layer (L3) header 206 includes a network address for the source node 102a-c (Fig. 1B), that is the IP source address 244, and a network address for the destination node 112a-c (Fig. 1B), that is, the IP destination address 246. Other fields in the IP network layer header 206 include Version 222, HLEN 224, Type of Service ("TOS") 226, Total Length 228, Identification 230, Flags 232, Fragment Offset 234, Time to Live ("TTL") 236, Protocol field 240, Header Checksum 242, Options 248 and pad 250.

A data packet 200 (Fig. 2A) received from a source node 102a-c (Fig. 1B) at an ingress port 138a-c (Fig. 1B) is bridged to one or more egress ports 136a-h (Fig. 1B) dependent on the destination address 214 (Fig. 2B) stored in the Ethernet data link (L2) header 204 (Fig. 2A) or is routed to one or more egress ports 136a-h (Fig. 1B) dependent on the IP destination address 246 stored in the IP network layer (L3) header 206.

Fig. 3 illustrates the forwarding logic 128 shown in Fig. 1B. The forwarding logic 128 includes a forward database 304. The forward database 304 selects a logical port forward vector 314 dependent on the contents of a search key 316. The forward database 304 is described in U.S. Patent No. 6,775,281, issued August 10, 2004.

The logical port forward vector 314 includes an egress port bit for each egress port 136a-h (Fig. 1B) in the switch 100 (Fig. 1B). An egress port bit is set to '1' to enable forwarding to the respective egress port 136a-h (Fig. 1B) and set to '0' to disable forwarding to the respective egress port 136a-h (Fig. 1B), thereby indicating all the egress ports to which a data packet can be forwarded. Thus, in a logical port forward vector 314 for destination 112c (Fig. 1B) egress port 2, egress port 3 and egress port 5 bits are set to '1' to enable the data packet to be forwarded to egress ports 136c, 136d and 136f (Fig. 1B) connected to destination 112c (Fig. 1B).

A trunk port selector vector 312 is selected by trunk port selector logic 302 dependent on a flow hash 310 generated by the flow hash logic 300. The trunk port selector vector 312 selects one of the egress ports (port 2, port 3 or port 5) 136c-e

-8-

enabled in the logical port forward vector 314, through which to forward a received data packet to destination 112c (Fig. 1B).

The trunk group membership table 318 includes a trunk group membership vector 320 for each ingress port 138 (Fig. 2) in the switch 100. The trunk group membership vector 320 includes a bit for each port in the switch 100. A bit is set to '1' in the port's trunk group membership vector indicating other ports which are in the same trunk group. If an ingress port is not a member of a trunk group, the ingress port's bit is the only bit set to '1' in the trunk group membership vector.

In a switch which supports trunking, a host may send and receive data packets on any of the ports associated with the trunk on which that host resides. It is necessary to identify which ports belong to the same trunk group on which the packet was received. As a result, ingress ports are used to index the trunk group membership table in order to ensure echo suppression, that is, ensure that an incoming packet is not forwarded to the same port or other ports of that particular trunk group. For example, if ingress port 0 is a member of a trunk group consisting of ports 0, 2 and 3, an incoming data packet can not be forwarded to ports 0, 2 or 3 because they are all part of the same trunk group. The trunk group membership table 318 stores this group membership information and ensures that such echoing will not occur.

The ingress port at which a data packet is received is forwarded on ingress port number 322 to the trunk group membership table 318. The ingress port number 322 is an index to a trunk group membership vector 320 in the trunk group membership table 318. The trunk group membership vector 320 is forwarded to the vector combine logic 306.

A trunk group membership vector 320 can be used to perform hardware learning by modifying, refreshing or adding a logical port forward vector 314 in the forward database 304. A learn port number 324 is stored with each logical port forward vector 314 stored in the forward data base. The learn port number 324 identifies the ingress port at which the source address was learned. The learn port number 324 is forwarded to the trunk group membership table 318. The trunk group membership table 318 determines if the logical port forward vector for the source address is to be modified dependent on the learn port number 324. If so, the trunk

group membership table forwards the updated logical port forward vector 326 to the forward database 304. The steps for determining if the logical port forward vector is to be updated are described in conjunction with Fig. 7.

The vector combine logic 306 combines the trunk port selector vector 312, the logical port forward vector 314, and the trunk group membership vector 320. The trunk port selector vector 312 provides a mask for the logical port forward vector 314, to select one of the enabled egress ports, through which to forward the received data packet to destination 112c.

A portion of the data packet's headers 308 are forwarded to the flow hash logic 300. The contents of the portion of the data packet's headers 308 is dependent on the network protocol encoded in the received data packet. If the received data packet includes a layer 3 header; for example, an IP network layer (L3) header 206 (Fig. 2C), the portion of the data packet's headers 308 includes the IP source address 244 (Fig. 2C), the IP destination address 246 (Fig. 2C), and the Protocol field 240 (Fig. 2C). If the data packet does not include a layer 3 header; for example, if the data packet is an Ethernet Protocol data packet, the portion of the data packet's headers 308 includes the L2 source address 216 (Fig. 2B) and the L2 destination address 214 (Fig. 2B). Thus, the contents of the portion of the data packet's header 308 identifies a data flow from a source 102a-c (Fig. 1B) to a destination 112a-c (Fig. 1B), so that data packets for the same flow (from a source to a destination) are forwarded through the same egress port 136a-h (Fig. 1B) on the same physical link 132c-e (Fig. 1B) to the destination 112c.

Fig. 4 is a flow diagram of the functions performed in the flow hash logic 300 shown in Fig. 3. The flow hash logic 300 (Fig. 3) generates a Cyclic Redundancy Check ("CRC") on the contents of a portion of the data packet's headers 308 (Fig. 1B). The contents of the portion of the data packet's headers 308 (Fig. 3) is selected from the data packet's headers and forwarded to the flow hash logic 300 (Fig. 3). The flow diagram is described in conjunction with Fig. 3.

At step 400, a CRC variable is initialized to an invalid value before the CRC is generated. The invalid value may be generated by setting all bits in the CRC variable to '1'. Processing continues with step 402.

At step 402, the flow hash logic 300 (Fig. 3) examines the contents of the L2 length/type field 220 (Fig. 2B) in the data link (L2) header 204 (Fig. 2B). If the length/type field 220 (Fig. 2B) is Internet Protocol Version 4 ("IPv4"), the data packet includes an IP network layer (L3) header 206 (Fig. 2C), and processing continues with step 404. If not, processing continues with step 410.

At step 404, the flow hash logic 300 (Fig. 3) examines the length field 228 (Fig. 2C) and the protocol field 240 in the IP network layer (L3) header 206 (Fig. 2C). If the length field 228 (Fig. 2C) contents are 'five' and the protocol field 240 stores User Datagram Protocol ("UDP") or Transport Control Protocol ("TCP"), processing continues with step 406. If not, processing continues with step 408.

At step 406, the flow hash logic 300 (Fig. 3) generates a CRC using the contents of the IP source address field 244 (Fig. 2C), the IP destination address field 246 (Fig. 2C) in the IP network layer (L3) header 206 (Fig. 2C), the contents of the L4 source port address field (not shown) and the contents of L4 destination port address field (not shown) in the transport layer (L4) header 208 (Fig. 2A) included in the data packet's headers. After the CRC is generated and stored in the CRC variable, processing continues with step 412.

At step 408, the flow hash logic 300 (Fig. 3) generates a CRC using the contents of the IP source address field 244 (Fig. 2C) and the IP destination address 246 (Fig. 2C) field in the IP network layer (L3) header 206 (Fig. 2C) included in the data packet's headers. After the CRC is generated and stored in the CRC variable, processing continues with step 412.

At step 410, the flow hash logic 300 (Fig. 3) generates a CRC using the contents of the L2 source address field 216 (Fig. 2B) and the L2 destination address field 214 (Fig. 2B) in the data link layer (L2) header 206 (Fig. 2B) included in the data packet's headers. After the CRC is generated and stored in the CRC variable, processing continues with step 412.

At step 412, the flow hash logic 300 (Fig. 3) selects the Least Significant Bits ("LSBs") of the stored CRC variable as flow hash bits 310 (Fig. 3). The flow hash bits 310 (Fig. 3) are forwarded to the trunk port selector table 302 (Fig. 3). Six flow hash bits 310 allow the selection of one of the sixty-four trunk port selector table entries stored in the truck port selector table 302 (Fig. 3).

-11-

By generating flow hash bits 310 dependent on the source and destination addresses included in the data packet's headers, data packets for the same flow (from a source 102 (Fig. 1B) to a destination 112 (Fig. 1B)) select the same trunk port selector table entry in the trunk port selector table 302 (Fig. 3). Also, by selecting a trunk port selector table entry dependent on a source address and a destination address, data flows are distributed amongst a plurality of physical links 132c-e (Fig. 1B) connected to a destination 112c (Fig. 1B).

Fig. 5 illustrates the trunk port selector table 302 shown in Fig. 3. The trunk port selector table 302 includes a plurality of trunk port selector table entries 500a-f. Six trunk port selector table entries 500a-f are shown in Fig. 5. The six flow hash bits 210 is an index to the sixty-four trunk port selector table entries, including 500a-f, stored in the trunk port selector table 302. Each trunk port selector table entry 500a-h includes a respective egress port bit 502a-h for each egress port 136a-h in the switch 100 (Fig. 1B).

Trunk port selector table entry 500a has eight egress port bits 502aa-ah; that is, an egress port bit 502aa-ah, for each egress port 136a-h (Fig. 1B) in the switch 100 (Fig. 1B). The invention is not limited to a switch with eight egress ports 136a-h (Fig. 1B), any number of egress ports 136 (Fig. 1B) may be supported by providing one egress port bit for each egress port in a trunk port selector table entry 500a-h. The state of the egress port bit 502 in the trunk port selector table entry 500 determines whether a data packet can be forwarded to the respective egress port 136a-h (Fig. 1B). In the embodiment shown in Fig. 5, a data packet may be forwarded to the egress port 136a-h (Fig. 1B) if the respective egress port bit 502 in the trunk port selector table entry 500a is '1'. In an alternative embodiment, a data packet may be forwarded to the egress port 136a-h (Fig. 1B) if the respective egress port bit 502 in the trunk port selector table entry 500a is '0'.

Egress port bits 502ac, 502ad and 502af associated with egress ports 136c, 136d, 136f (Fig. 1B) are members of trunk group 134 (Fig. 1B). In each trunk port selector table entry 500, one of the members of the trunk group 134 (Fig. 1B) is enabled with the respective egress port bit 502 set to '1'. In trunk port selector table entry 500a egress port bit 502ac is '1' to enable egress port 2 136c (Fig. 1B) in the logical link 134 (Fig. 1B). In trunk port selector table entry 500b, egress port bit

-12-

502bd is '1' to enable egress port 3 136d (Fig. 1B) in the logical link 134 (Fig. 1B). In the embodiment shown in Fig. 5, egress port 5 136f (Fig. 1B) is enabled in trunk port selector table entry 500c, egress port 2 136c (Fig. 1B) is enabled in trunk port selector table entry 500d, egress port 3 136d (Fig. 1B) is enabled in trunk port selector table entry 500e and egress port 5 136f (Fig. 1B) is enabled in trunk port selector table entry 500f.

The distribution of the port selector table entries 500 in the port selector table determines the number of data packets forwarded through each of the physical links in a logical link. If all physical links transmit data packets at the same speed, data packets may be evenly distributed, dependent on the number of port selector table entries 500 enabling each of the physical links in the logical link.

Alternatively, a greater percentage of the data packets may be forwarded on a particular physical link in the logical link dependent on the number of port selector entries 500 enabling the physical link in the logical link. A greater percentage of data packets for a destination connected to a switch by a logical link may be forwarded on a physical link which is faster than the other physical links in the logical link.

For example, if physical link 132c is 1G bits per second link and physical links 132d-e are 100Mbits per second links, the port selector table entries 500 stored in the trunk port selector table 302 may be generated such that 80% of the data packets are forwarded on physical link 132 c and 10% on each of physical links 132d-e.

Thus, the proportion of data packets transmitted on a particular physical link in a logical link is dependent on the distribution of port selector table entries 500 in the port selector table 302.

Fig. 6 illustrates the combination of one of the trunk port selector entries 500 shown in Fig. 5, a logical port forward vector entry 600, and a trunk group membership vector entry 604 for a data packet received from source port 102a (Fig. 1) for destination 112c (Fig. 1). Trunk port selector entry 500a is forwarded on trunk port selector vector 312 to vector combine logic 306. Logical port forward vector entry 600 is forwarded on logical port forward vector 314 to vector combine logic 306. Trunk group membership vector entry 604 is forwarded on trunk group membership vector 320 to vector combine logic 306.

-13-

The trunk group membership vector entry 604 is the vector selected for source port 102a (Fig. 1). Source port 102a (Fig. 1) is connected to ingress port 0 138a which in this example, is a single physical link. Thus, bit 604a is set '1' indicating the port to which source port 102a is connected.

Logical port forward vector entry 600 has bits 600f, 600d, 600c set to '1' to enable data packets to be forwarded to egress ports 136c, 136d, 136f (Fig. 1B); that is, all the egress ports 136 in the logic link 134. All other bits 600g, 600h, 600e, 600a, 600b are '0' disabling forwarding of data packets to the respective egress ports 136 (Fig. 1B). Trunk port selector entry 500a has bit 502ac set '1' to select egress port 2 136c (Fig. 1B).

The vector combine logic 306 inverts the trunk group membership vector entry 604 and performs a bit-wise logical AND function to combine the trunk port selector entry 500a, and the inverted trunk group membership vector entry 604. The forward vector entry 602 resulting from the combination has bit 602c set to '1' enabling data packets for the data flow to be forwarded to egress port_2 136c (Fig. 1B).

The trunk group membership vector 604 is inverted in order to provide echo suppression support, that is, by disabling forwarding of a data packet on a logical link if the data packet was received on the logical link. For example, a data packet received on ingress port 2 138c is not forwarded on egress port 5 136f because ingress port 2 138c and egress port 5 136f are members of the same logical link 134.

Thus the logical port forward vector 314 enables all egress ports in the logical link 134 (Fig. 1B) and the trunk port selector vector 312 selects one of the physical links in the logical link 134 (Fig. 1B). The vector combine logic 306 generates the forward vector 114 by combining the trunk port selector vector 312, the trunk group membership vector 320 and the logical port forward vector 314.

Fig. 7 is a flow diagram of the steps for using the contents of a trunk group membership vector 320 to modify a logical port forward vector 314 stored in the forward database 304. The steps are described in conjunction with the block diagram shown in Fig. 3

At step 700, a search key 316 including a source address included in a received data packet is forwarded to the forward data base. Also, the ingress port

-14-

number 322 at which the received data packet was received is forwarded to the trunk group membership table 318. Processing continues with step 702.

At step 702, the forward database determines if there is an entry corresponding to the search key 316. If so, processing continues with step 704. If not, processing continues with step 708.

At step 704, the learn port number 324 stored at the forward database entry corresponding to the search key 316 is forwarded to the trunk group membership table 318. The learn port number 324 is the ingress port at which the source address included in the received data packet was learned. Processing continues with step 706.

At step 706, the trunk group membership table 318 checks to see if the bit corresponding to the learn port number 324 is set to '1' in the trunk group membership vector 320 for source port number 322. If so, processing is complete because the received data packet was not received on a new port. If not, processing continues with step 708 .

At step 708, the received data packet arrived on an ingress port other than the ingress port on which it was first learned or it is the first time that a data packet has been received from the source address. Thus, the logical port forward vector 314 may need to be updated with the ingress port number 322 and any other ports which are members of the same trunk group as the ingress port number 322. The trunk group membership vector 320 corresponding to the ingress port number 322 is forwarded on update 326 so that the forward database 304 can modify or refresh the logical port forward vector 314 corresponding to the source address. Processing is complete.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

CLAIMS:

1. A switch comprising:
 - a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports in the switch;
 - flow hash logic which selects a flow hash for a received data packet dependent on the result of a function performed on a destination address and a source address included in the received data packet, the destination address and source address selected dependent on the contents of a type field in a data link header; and
 - trunk port selector logic which selects a trunk port entry dependent on the flow hash, the trunk port entry providing one of the plurality of physical links on which to forward the received data to the destination.
2. A switch as claimed in claim 1 wherein the plurality of ports are randomly selected from the available ports in the switch.
3. A switch as claimed in claim 1 or 2 wherein the destination address is an Ethernet destination address and the source address is an Ethernet source address.
4. A switch as claimed in claim 1, 2, or 3 wherein the destination address is an internet protocol (IP) destination address and the source address is an IP source address.
5. A switch as claimed in any one of claims 1 to 4 wherein the destination address is a transport control protocol (TCP) destination port address and the source address is a TCP source port address.
6. A switch as claimed in any one of claims 1 to 5 wherein the destination address is a user datagram protocol (UDP) destination port address and the source address is a UDP source port address.

7. A switch as claimed in any one of claims 1 to 6 wherein data packets for a same flow from the source address to the destination address are forwarded on a same physical link through a same egress port.
8. A switch comprising:
 - a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports in the switch;
 - flow hash logic which indexes a flow hash for received data dependent on a destination address and a source address included in the received data packet;
 - trunk port selector logic which selects a trunk port entry dependent on the flow hash, the trunk port entry providing one of the plurality of physical links on which to forward the received data to the destination; and
 - vector combine logic which selects a port in the switch to forward the received data on one of the physical links in the logical link, dependent on a combination of a logical port forward vector and the trunk port entry.
9. A switch as claimed in claim 8 wherein the vector combine logic selects a port dependent on a trunk group membership vector.
10. A switch as claimed in claim 8 or 9 wherein the port is not selected if the received data was received on any one of the physical links in the logical link.
11. A switch comprising:
 - a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports;
 - means for selecting a flow hash for a received data packet dependent on the result of a function performed on a destination address and a source address included in the received data packet, the destination address and the source address selected dependent on the contents of a type field in a data link header; and

means for selecting a trunk port entry dependent on the flow hash, the trunk port entry selecting one of the plurality of physical links on which to forward the received data to the destination.

12. A switch as claimed in claim 11 wherein the plurality of ports are randomly selected from the available ports in the switch.
13. A switch as claimed in claim 11 or 12 wherein the destination address is an Ethernet destination address and the source address is an Ethernet source address.
14. A switch as claimed in claim 11, 12, or 13 wherein the destination address is an internet protocol (IP) destination address and the source address is an IP source address.
15. A switch as claimed in any one of claims 11 to 14 wherein the destination address is a transport control protocol (TCP) destination port address and the source address is a TCP source port address.
16. A switch as claimed in any one of claims 11 to 15 wherein the destination address is a user datagram protocol (UDP) destination port address and the source address is a UDP source port address.
17. A switch as claimed in any one of claims 11 to 16 wherein data packets for a same flow from the source address to the destination address are forwarded on a same physical link through a same egress port.
18. A switch comprising:
 - a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports;
 - means for selecting a flow hash for received data dependent on a destination address and a source address included in the received data;

-18-

means for selecting a trunk port entry dependent on the flow hash, the trunk port entry selecting one of the plurality of physical links on which to forward the received data to the destination; and

means for selecting a port to forward the received data on one of the physical links in the logical link, dependent on a logical port forward vector and the trunk port entry.

19. A switch as claimed in claim 18 wherein the vector combine logic selects a port dependent on a trunk group membership vector.
20. A switch as claimed in claim 19 wherein the port is not selected if the received data was received on any one of the physical links in the logical link.
21. A method for forwarding received data through a switch comprising the steps of:
 - providing a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports;
 - selecting by flow hash logic, a flow hash for the received data packet dependent on the result of a function performed on a destination address and a source address included in the received data, the destination address and the source address selected dependent on the contents of a type field in a data link header;
 - selecting, by trunk port selector logic, a trunk port entry dependent on the flow hash; and
 - providing by the trunk port entry, one of the plurality of physical links on which to forward the received data to the destination.
22. A method as claimed in claim 21 wherein the plurality of ports are randomly selected from the available ports in the switch.
23. A method as claimed in claim 21 or 22 wherein the destination address is an Ethernet destination address and the source address is an Ethernet source address.

24. A method as claimed in claim 21, 22, or 23 wherein the destination address is an internet protocol (IP) destination address and the source address is an IP source address.
25. A method as claimed in any one of claims 21 to 24 wherein the destination address is a transport control protocol (TCP) destination port address and the source address is a TCP source port address.
26. A method as claimed in any one of claims 21 to 25 wherein the destination address is a user datagram protocol (UDP) destination port address and the source address is a UDP source port address.
27. A method as claimed in any one of claims 21 to 26 wherein data packets for a same flow from the source address to the destination address are forwarded on a same physical link through a same egress port.
28. A method comprising the steps of:
 - providing a logical link connecting a destination to the switch, the logical link comprising a plurality of physical links connected to a plurality of ports;
 - selecting by flow hash logic, a flow hash for the received data dependent on a destination address and a source address included in the received data;
 - selecting, by trunk port selector logic, a trunk port entry dependent on the flow hash;
 - providing by the trunk port entry one of the plurality of physical links on which to forward the received data to the destination; and
 - selecting, by vector combine logic, a port in the switch to forward the received data on one of the physical links in the logical link dependent on a combination of a logical port forward vector and the trunk port entry.
29. A method as claimed in claim 28 wherein the vector combine logic selects a port dependent on a trunk membership vector.

30. A method as claimed in claim 29 wherein if the port is not selected if the received data was received on any one of the physical links in the logical link.
31. A data network switch comprising:
 - flow hash logic which generates a flow hash dependent on source and destination information included in data received by the data network switch;
 - a forward database for storing logical port forward vectors, each logical port forward vector including forwarding information;
 - a trunk group membership table for storing trunk group membership vectors, each trunk group membership vector indicating which ports belong to a trunk group; and
 - vector combine logic which combines the logical port forward vector, the trunk port select vector and the trunk group membership vector to select a single port on which to forward to the received data.
32. A data network switch as claimed in claim 31 wherein the vector combine logic selects the single port dependent on the trunk group membership vector associated with the port on which the data was received.
33. A data network switch as claimed in claim 32 wherein the single port is not selected if the received data was received on the single port or on any other members of the trunk group to which the single port belongs.
34. A data network switch as claimed in claim 31, 32, or 33 wherein a trunk group membership vector is used to modify a corresponding logical port forward vector.
35. A data network switch as claimed in any one of claims 31 to 34 wherein the trunk group membership vector is used to add a logical port forward vector to the forward database.

-21-

36. A data network switch as claimed in any one of claims 31 to 35 wherein data packets for a same flow from the source address to the destination address are forwarded on a same physical link through a same egress port.

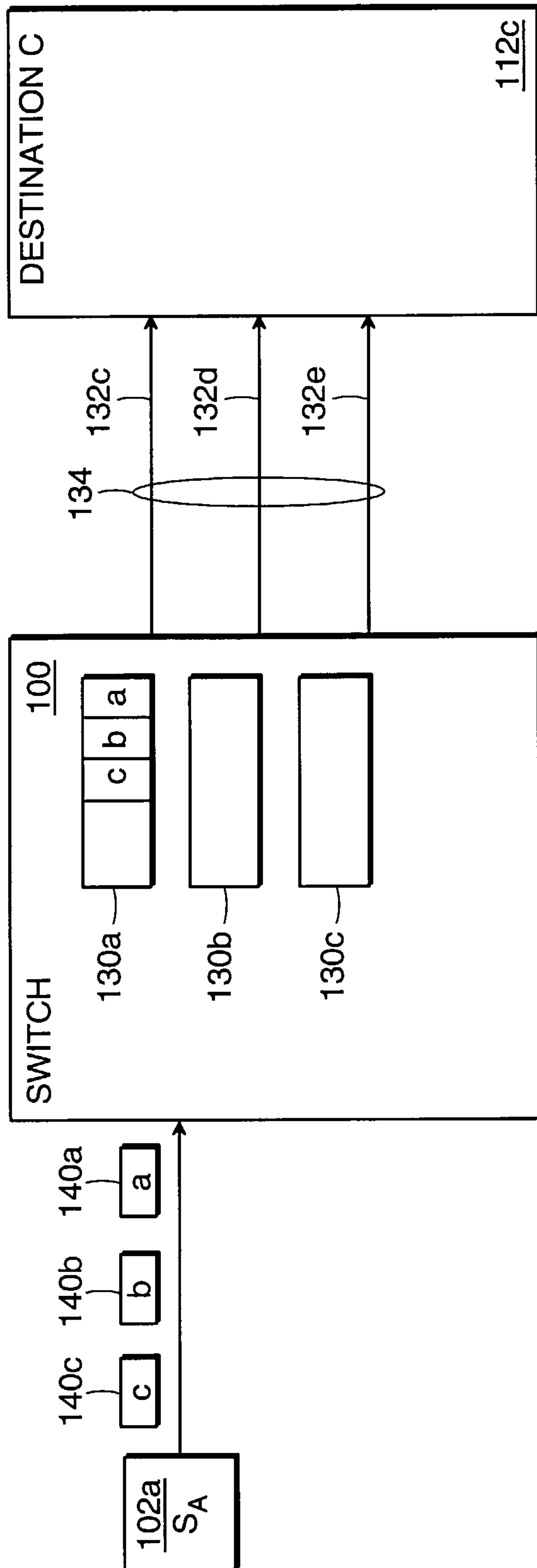


FIG. 1A

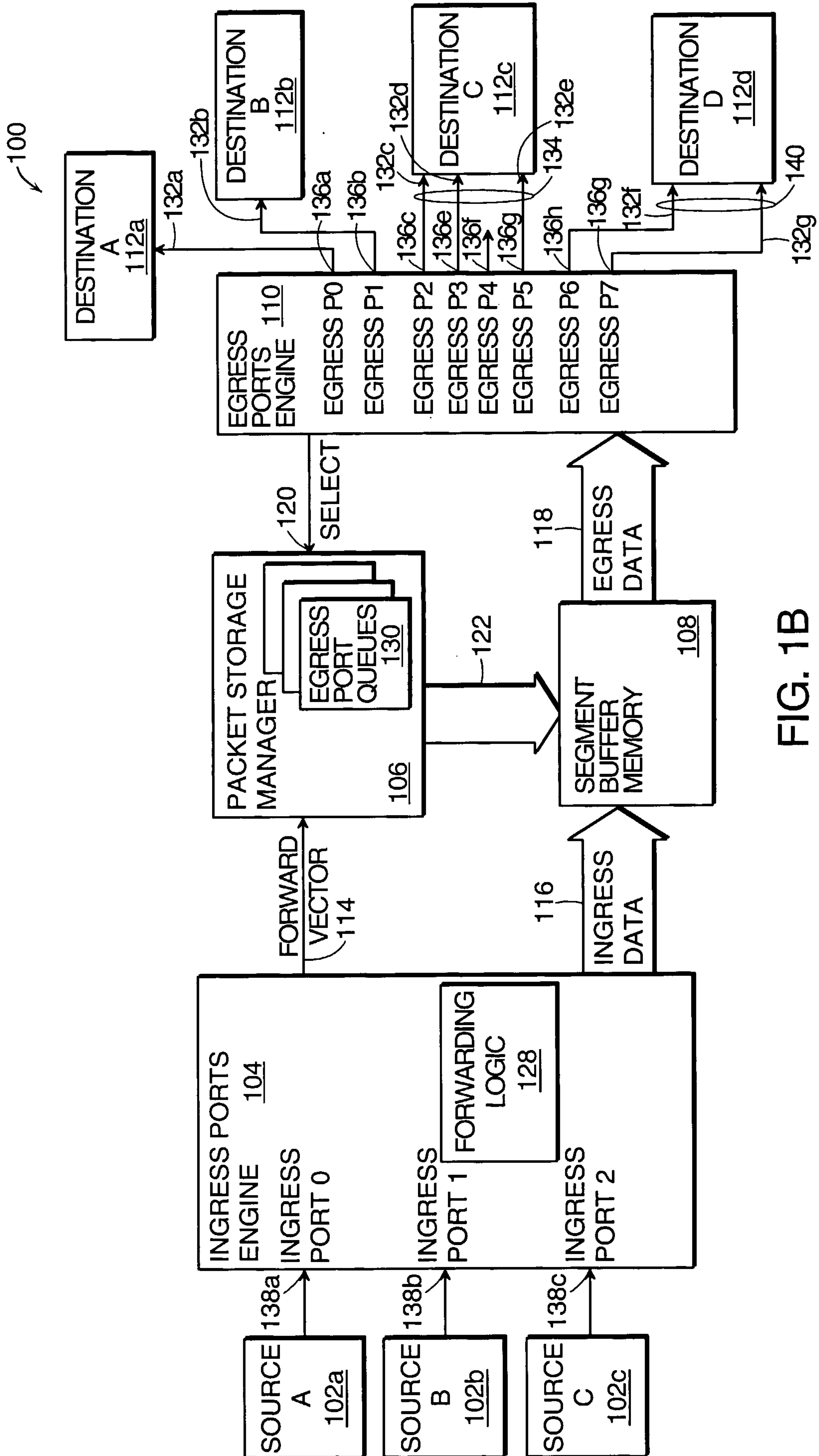
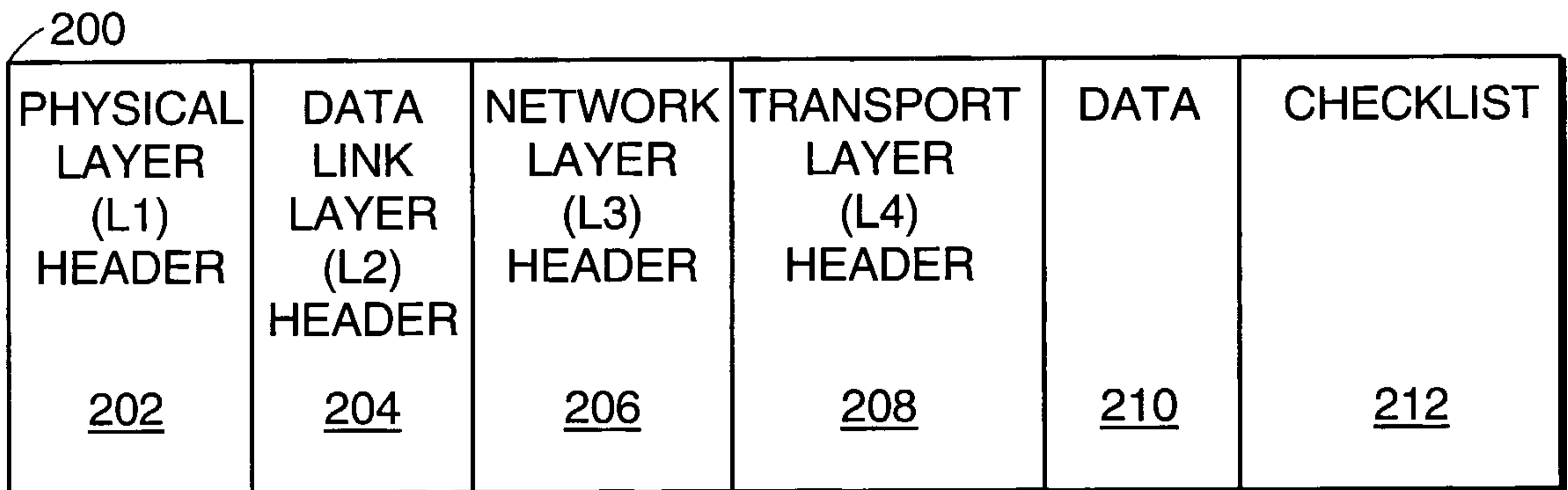
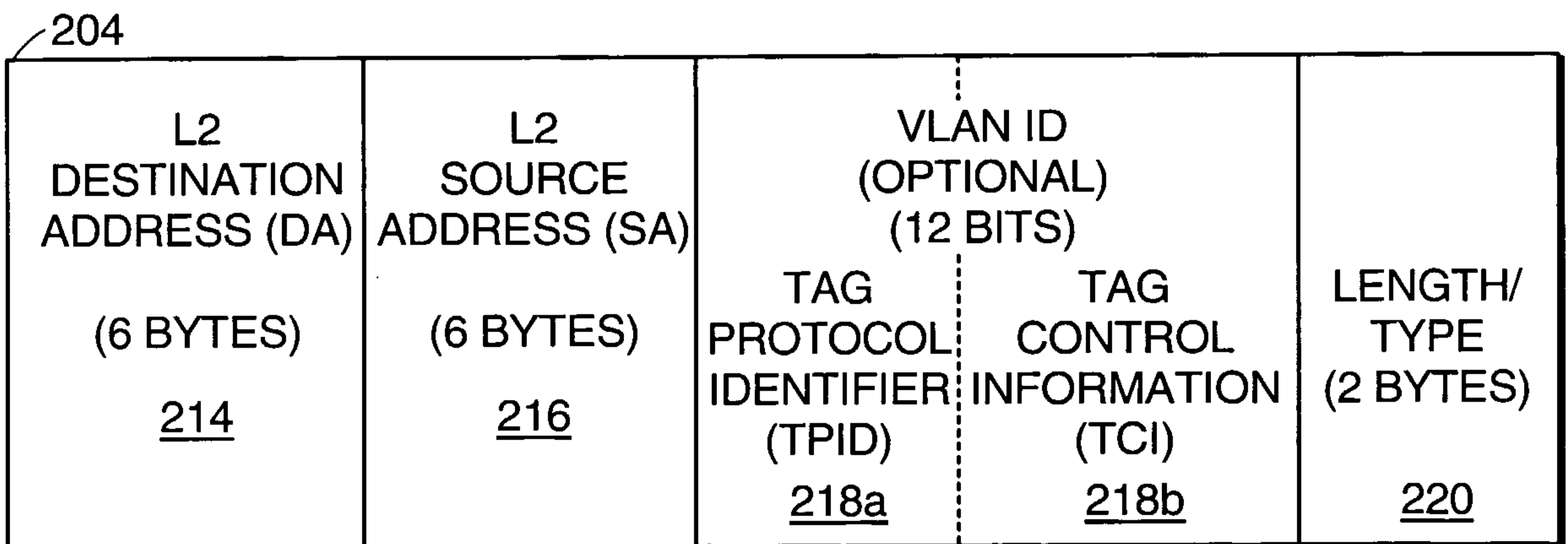


FIG. 1B



PRIOR ART
FIG. 2A



PRIOR ART
FIG. 2B

4/9

<u>206</u>			
<u>222</u> VERS	<u>234</u> HLEN	<u>236</u> TOS	<u>228</u> TOTAL LENGTH
<u>230</u> IDENTIFICATION		<u>232</u> FLAGS	<u>234</u> FRAGMENT OFFSET
<u>236</u> TTL	<u>240</u> PROTOCOL	<u>242</u> HEADER CHECKSUM	
IP SOURCE ADDRESS			<u>244</u>
IP DESTINATION ADDRESS			<u>246</u>
<u>248</u> OPTIONS			<u>250</u> PAD

PRIOR ART
FIG. 2C

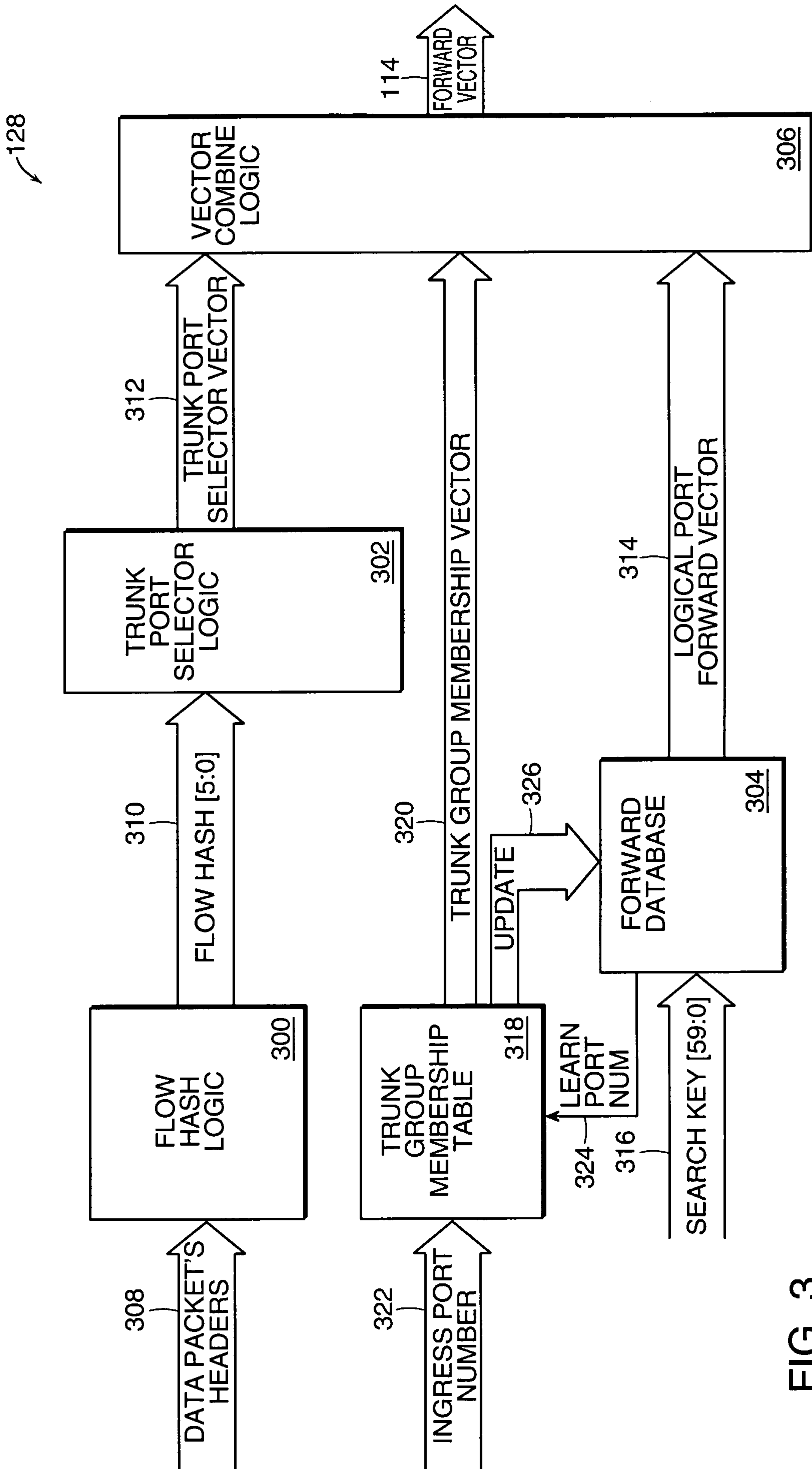


FIG. 3

6/9

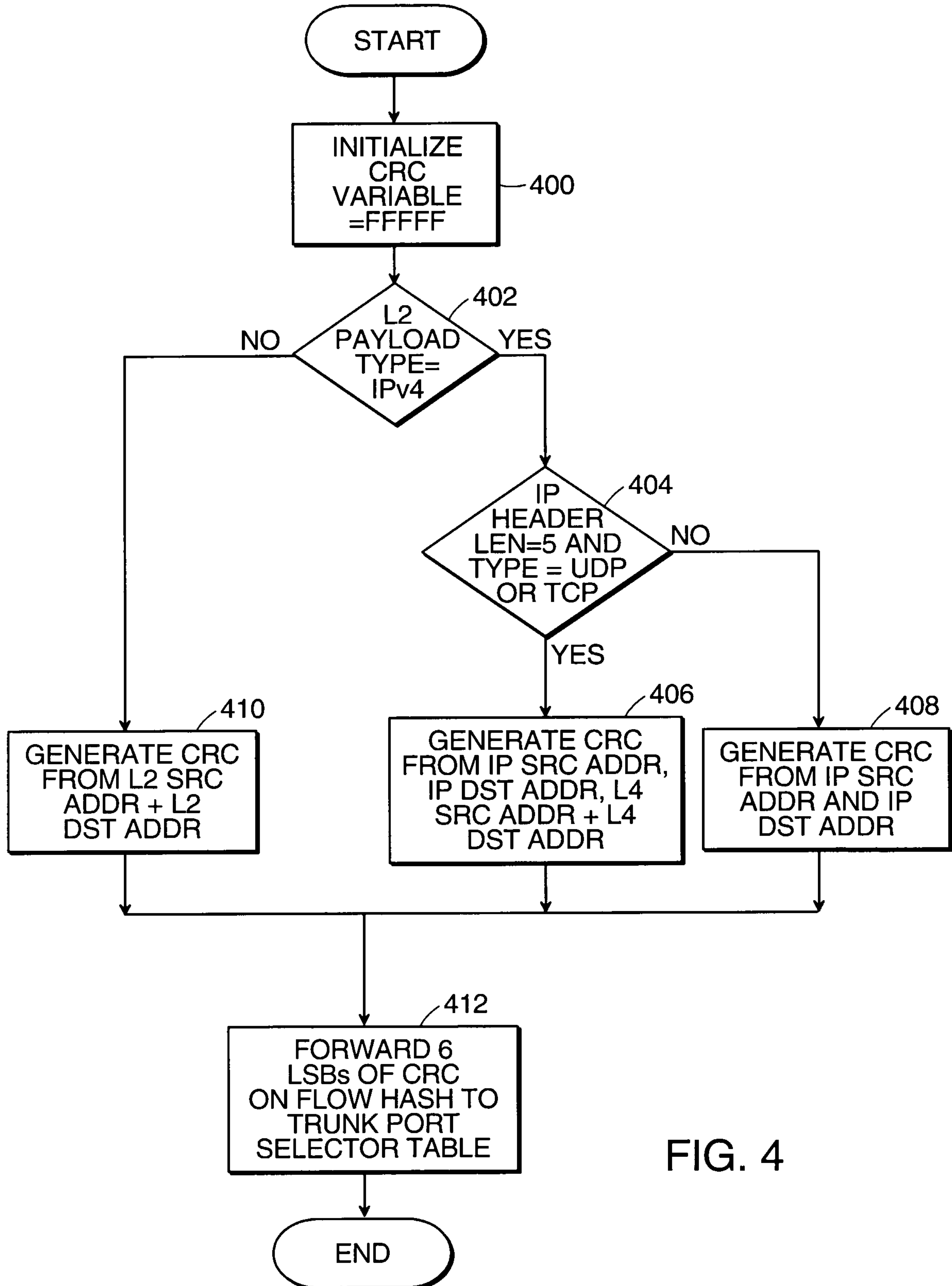


FIG. 4

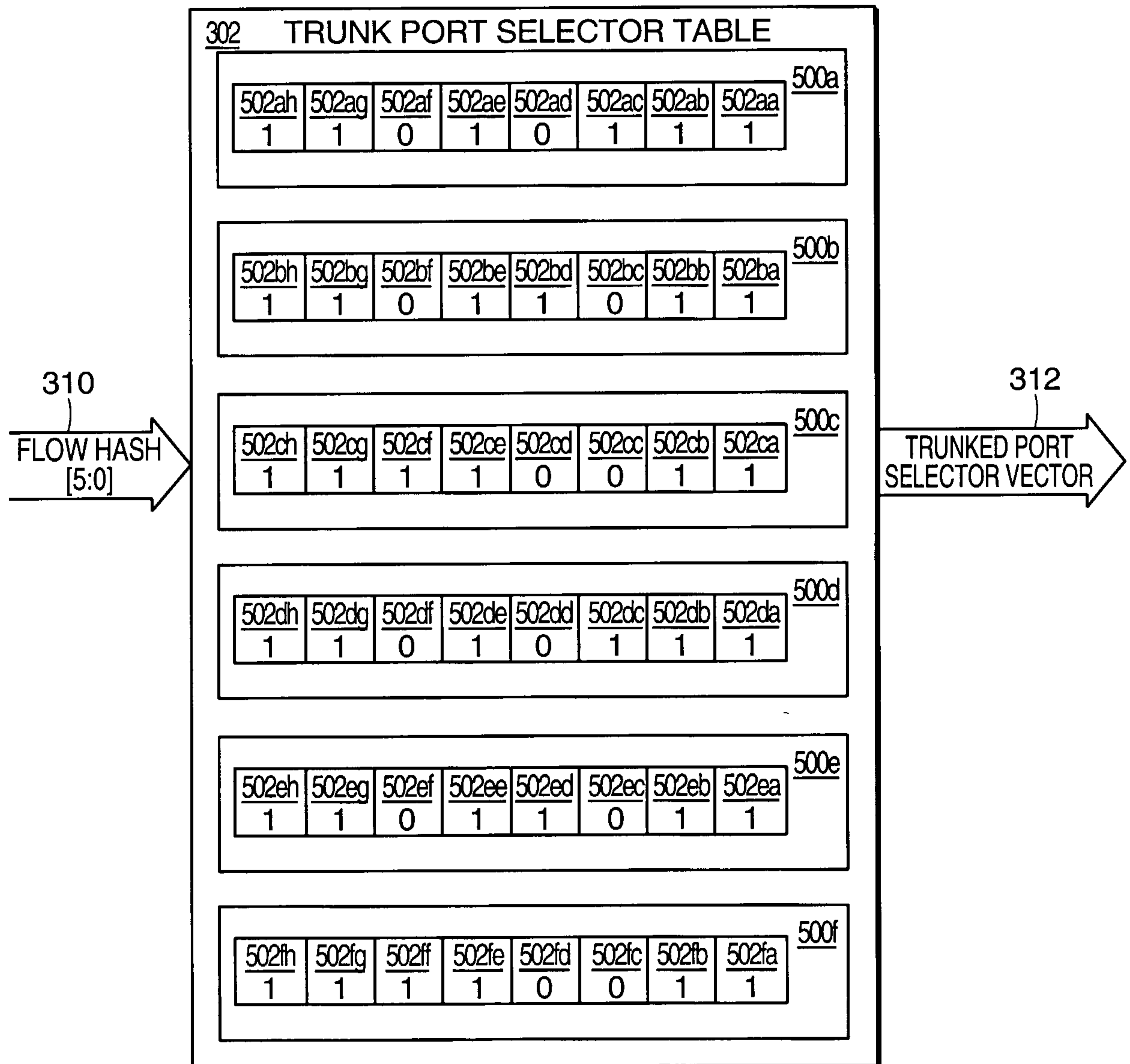


FIG. 5

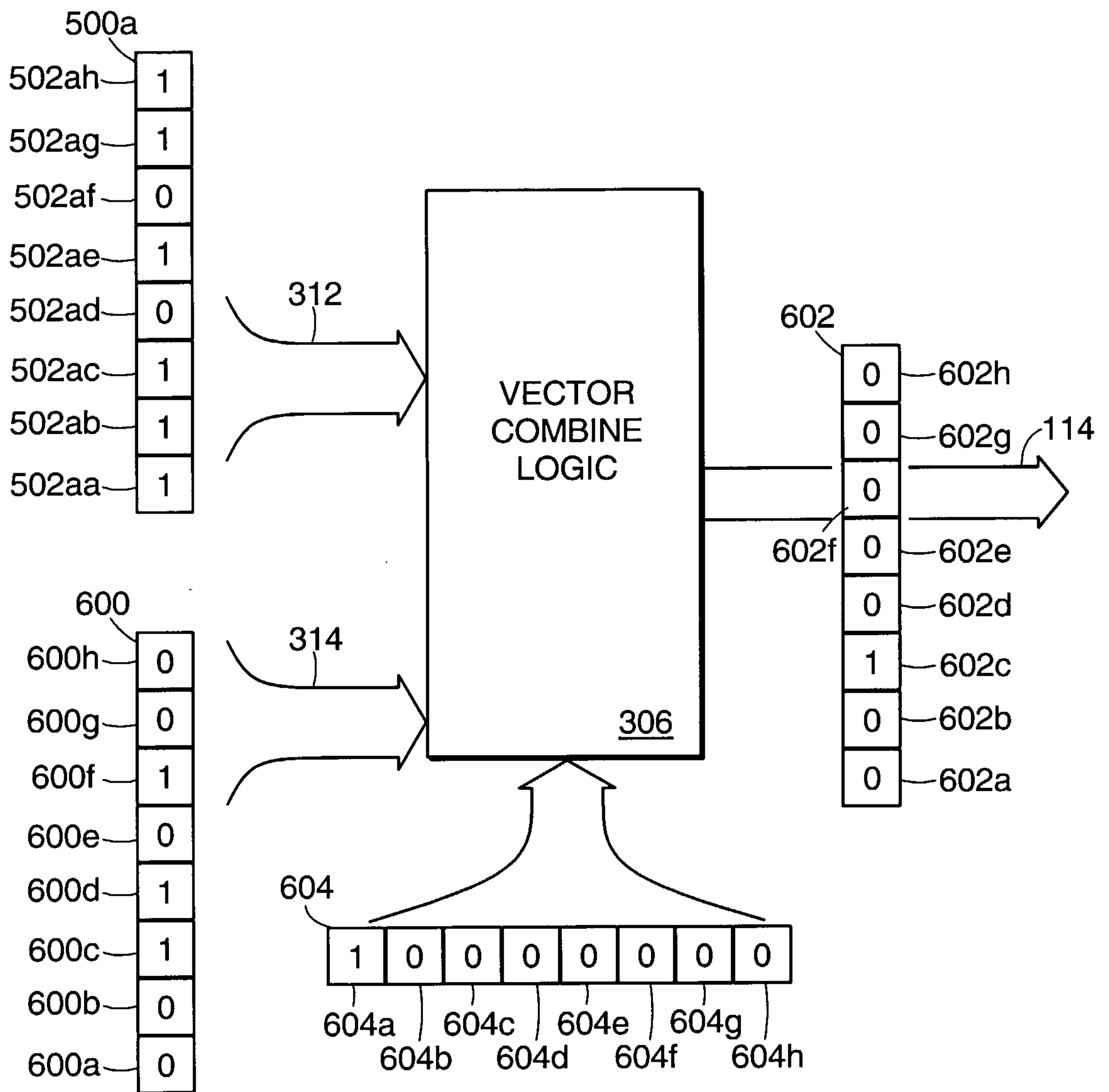


FIG. 6

9/9

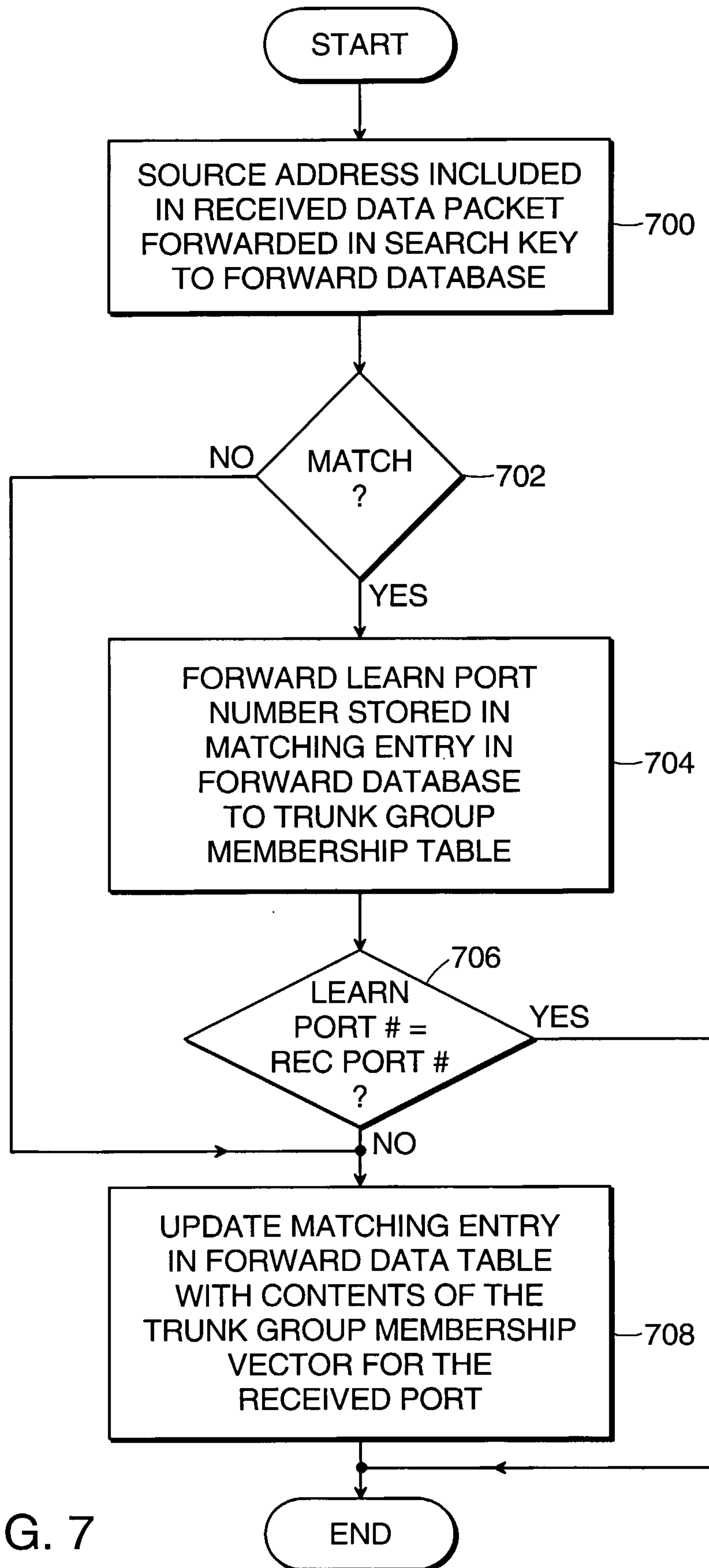


FIG. 7

