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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 September 2012 (27.09.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/129432 A2

(51) International Patent Classification:
H04L 12/56 (2006.01)

(21) International Application Number:
PCT/US2012/030175

(22) International Filing Date:
22 March 2012 (22.03.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/466,232 22 March 2011 (22.03.2011) US
13/323,594 12 December 2011 (12.12.2011) US

(71) Applicant *(for all designated States except US)*: **TEXAS INSTRUMENTS INCORPORATED** [US/US]; P.O. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).

(71) Applicant *(for JP only)*: **TEXAS INSTRUMENTS JAPAN LIMITED** [JP/JP]; 24-1, Nishi-shinjuku 6-chome, Shinjuku-ku Tokyo, 160-8366 (JP).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BHADRA, Sandeep** [US/US]; 2401 Bennett Ave., Dallas, TX 75026 (US). **KOKRADY, Aman, A.** [IN/IN]; 101D Balaji Sapthagiri Blck Ii, Kundanhali, Vlg, Kr Puram Hobli, B'lore East Mahadevapura, Cmc, Bangalore 560037 (IN). **BOSSHART, Patrick, W.** [US/US]; 7508 Tara Court, Plano, TX 75025 (US). **KIM, Hun-seok** [KR/US]; 15575 Lewis Place, Apt. 3645, Addison, TX 75001 (US).

(74) Agents: **FRANZ, Warren, L.** et al.; Texas Instruments Incorporated, Deputy General Patent Counsel, P.o. Box 655474, Mail Station 3999, Dallas, TX 75265-5474 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR PACKET SWITCHING

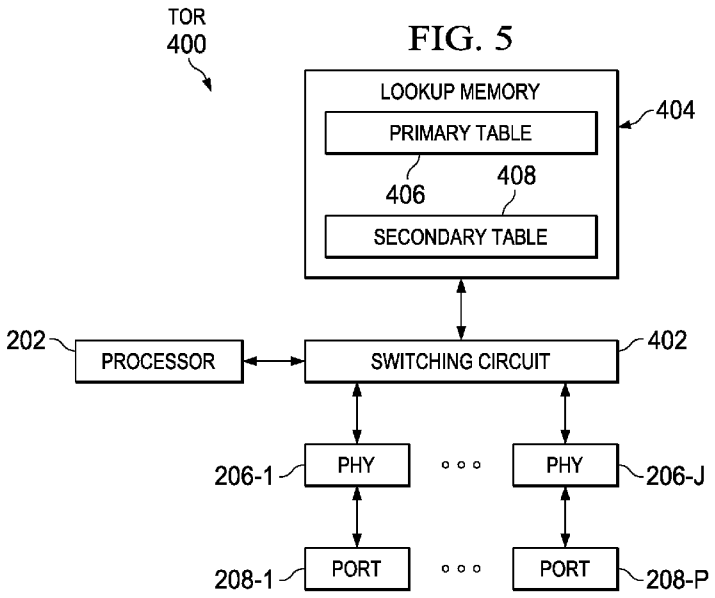


FIG. 5

(57) Abstract: A method for processing packet lookups is provided. Packets (which each have a body and a header) are received and parsed to parsing headers. A hash function is applied to each header, and each hashed header is compared with a plurality of binary rules stored within a primary table (406), where each binary rule is a binary version of at least one ternary rule from a first set of ternary rules. For each match failure with the plurality of rules, a secondary table (408) is searched using the header associated with each match failure, where the secondary table (408) includes a second set of ternary rules.

WO 2012/129432 A2

DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

METHOD AND APPARATUS FOR PACKET SWITCHING

[0001] This relates generally to packet switching and, more particularly, to top-of-rack (TOR) switches.

BACKGROUND

[0002] FIG. 1 illustrates a conventional routing model for packet switching networks can be seen. In this model, a core network 112 communicates over the internet 102 through core routers 104-1 to 104-N. The core network 112 generally communicates with the core routers 104-1 to 104-N with intermediate switches 106-1 to 106-M; usually, two intermediate switches (i.e., 106-1 and 106-2) communicate with a core router (i.e. 104-1). The intermediate switches 106-1 to 106-M are then each able to communicate with each aggregate switch 108-1 to 108-K, which are each in communication with TOR switches 110-1 to 110-L. These TOR switches 110-1 to 110-L can then each be in communication with several (i.e., 20) servers.

[0003] Of interest here are the TOR switches 110-1 to 110-K, and a diagram of an example of a TOR switch (which is labeled 110) can be seen in FIGS. 2 and 3. Usually, as part of a data center, servers are held in a "rack" and the TOR switch 110 is located within the rack (typically at the top) so as to operate as a forwarding switch. As shown, this TOR switch 110 is generally comprised of a processor 202, switching circuit 204, a ternary context-addressable memory (TCAM) 210, and input/output (I/O) circuitry (which generally includes physical layer (PHY) circuit 206-1 to 206-J and ports 208-1 to 208-P), and the switching circuit 204 generally comprises an input queue 302, a parser 304, a search engine 306, processor interface 308, action circuit 310, and output queue 312. In operation, data packets (which each generally have a header and a body) are received through the ports 208-1 top 208-P and PHY circuits 206-1 to 206-J. These packets are stored in the input queue 302 (which is typically a first-in-first-out (FIFO) circuit), and the parser 304 is able to extract the header from these queued packets. Using the extracted headers, the search engine 210 is able to search the TCAM 210 to determine a rule associated with the header, where each rule

is associated with an action. Once identified, the action circuit modifies the packet (usually the header) in accordance with the action associated identified rule. The modified packet is then placed in the output queue 312 (which is typically a FIFO circuit) so as to be transmitted.

[0004] The search engine 306 performs packet lookups using the TCAM 210, which is a high speed memory that allow for matches over a large database of ternary packet-forwarding rules (i.e., Access Control Lists, Destination IP rules, and NetFlow rules). TCAM 210, though, consume several multiples of power and area compared to other memory types (such as SRAM or embedded DRAM) making it difficult to embed large TCAMs on-chip. As a result, TOR switches 110-1 to 110-L suffer from in penalties of power and area, as well as limited flexibility because the TOR switches 110-1 to 110-L set the forwarding rules. Therefore, there is a need for an improved TOR switch with a lower cost and higher flexibility.

[0005] Some other conventional systems are described in: US Patent Nos. 7,028,098; 7,234,019; and 7,382,787; US Patent Publ. Nos. 2005/0262294 and 2011/0161580; and Mysore et al., "PortLand: A Scalable Fault-Tolerant layer 2 Data Center Network Fabric," SIGCOMM 2009, August 17-21, 2009.

SUMMARY

[0006] An example embodiment provides an apparatus. The apparatus comprises a lookup memory having a primary table and a secondary table, wherein the secondary table includes a first set of ternary rules, and wherein the primary includes a set of binary rule, and wherein each binary rule is a binary version of at least one ternary rule from a second set of ternary rules; and a search engine that is coupled to the lookup memory, wherein the search engine includes: an controller that is configured to receive data words; and hash logic that is coupled to lookup memory and the controller, wherein the hash logic is configured to perform a binary search of the primary table to determine whether each data word matches at least one of the binary rules, and wherein, if there is a failed match by hash logic and primary table, the search engine is configured to perform a ternary search of the secondary table to determine whether the data word associated with the failed match matches at least one of the ternary rules from the first set of ternary rules.

[0007] In accordance with an embodiment, the primary table further comprises: a dynamic memory; and stash.

[0008] In accordance with an embodiment, the stash is a content-addressable memory (CAM).

[0009] In accordance with an embodiment, the dynamic memory is a static random access memory (SRAM).

[0010] In accordance with an embodiment, the secondary table further comprises a Ternary CAM (TCAM).

[0011] In accordance with an embodiment, the apparatus further comprises: a shared memory; a plurality of port managers, wherein each port manager includes: an communication circuitry that is configured to receive input data packets and that is coupled to the shared memory and the search engine; and a parser that is coupled to the communication circuitry, wherein the parser is configured to parse each input data packet and extract its header, wherein each data word is associated with at least one header.

[0012] In accordance with an embodiment, the apparatus further comprises an action table that is in communication with the search engine.

[0013] In accordance with an embodiment, the communication circuitry further comprises: a media access controller (MAC) that is coupled to the parser; a transmit pipeline that is coupled between the shared memory and the MAC; a receive pipeline that is coupled between the shared memory and the MAC; and a search interface that is coupled between the parser and the search engine.

[0014] In accordance with an embodiment, the hash logic applies a keyed hash function to each data word.

[0015] In accordance with an embodiment, a method is provided. The method comprises receiving a plurality of packets, wherein each packet has a body and a header; parsing each packet to extract its header; applying a hash function to each header; comparing each hashed header with a plurality of binary rules stored within a primary table, wherein each binary rule is a binary version of at least one ternary rule from a first set of ternary rules; and for each match failure with the plurality of rules, searching a secondary table using the header associated with each match failure, wherein the secondary table includes a second set of ternary rules.

[0016] In accordance with an embodiment, the step of searching the secondary table further comprises simultaneously searching a plurality of banks within the TCAM.

[0017] In accordance with an embodiment, the method further comprises: generating a new rule and a new action for each match failure; and storing the new rule and new action in the SRAM.

[0018] In accordance with an embodiment, the hash function is a keyed hash function.

[0019] In accordance with an embodiment, an apparatus is provided. The apparatus comprises a primary table including a set of binary rule, and wherein each binary rule is a binary version of at least one ternary rule from a first set of ternary rules; a secondary table including a first set of ternary rules; a switching circuit having: a shared memory; a search engine including: an controller that is configured to receive data words; and hash logic that is coupled to lookup memory and the controller, wherein the hash logic is configured to perform a binary search of the primary table to determine whether each data word matches at least one of the binary rules, and wherein, if there is a failed match by hash logic and primary table, the search engine is configured to perform a ternary search of the secondary table to determine whether the data word associated with the failed match matches at least one of the ternary rules from the first set of ternary rules; and a plurality of port managers that are each in communication with the search engine; and input/output (I/O) circuit that is in communication with the switching circuit.

[0020] In accordance with an embodiment, the I/O circuitry further comprises: a plurality of physical layer (PHY) circuits, wherein each PHY circuit is in communication with the switching circuit; and a plurality of ports, wherein each port is in communication with at least one of the PHY circuits.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] Example embodiments are described with reference to accompanying drawings, wherein:

[0022] FIG. 1 illustrates a conventional routing model for a switched packet network;

[0023] FIG. 2 illustrates a conventional TOR switch of FIG. 1;

[0024] FIG. 3 illustrates the switching circuit from the TOR switch of FIG. 2;

[0025] FIG. 4 illustrates an example of a routing model in accordance with an embodiment of principles of the invention;

[0026] FIG. 5 illustrates an example of a TOR switch of FIG. 4;

[0027] FIG. 6 illustrates an example of a switching circuit from the TOR switch of FIG. 5;

[0028] FIG. 7 illustrates an example of the port manager of the switching circuit of FIG. 6

[0029] FIG. 8 and 9 illustrate examples of the search engine and lookup memory of FIGS. 5 and 6;

[0030] FIG. 10 illustrates an example of packet descriptor;

[0031] FIG. 11 illustrates an example of the action table of FIG. 5;

[0032] FIG. 12 illustrates an example of a lookup descriptor; and

[0033] FIG. 13 illustrates an example of a buffer descriptor.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

[0034] To increase network flexibility, a new Ethernet networking standard has been developed. This standard is known as the OpenFlow protocol, and version 1.1.0 (which was released on February 28, 2011) by the OpenFlow Switch Consortium is incorporated herein by reference for all purposes. In FIG. 4, an example of a routing model for this new standard can be seen. As shown, this model is similar to the model shown in FIG. 1, except that there is a network controller 401 that is able to control TOR switches 400-1 to 400-L. Network controller 401 may control other features (such as some within aggregate switches) but those control are omitted here for the sake of simplicity. With this configuration, the network controller 401 can set forwarding rules, while the TOR switches 400-1 to 400-L perform switching. This allows data centers to implements their own routing and flow management protocols.

[0035] Heavy reliance of TCAMs can be avoided with TOR switches 400-1 to 400-L, but attempting to design data-structures in hardware memory to reduce reliance on TCAMs can be difficult. Systems employing such architectures can be inefficient (wasting more memory than is used to store real addresses), hence, to be able to implement this, TOR switches 400-1 to 400-L (labeled 400 in FIG. 5) use a switching circuit 402 (which is typically an integrated circuit or IC) that is able to communicate with a lookup memory 404

(which generally has a primary table 406 and secondary table 408). The secondary table 408 generally stores ternary entries, while the primary table 406 generally stores binary entries. This allows the switching circuit 402 (which is shown in detail in FIGS. 6-13) to perform "primary" searches for more common searching events using the primary table 406 and, when the "primary" search fails, to perform "secondary" searches with the secondary table 408.

[0036] Turning first to the port managers 508-1 to 508-J of switching circuit 402, an example implementation can be seen in FIG. 7 (which is labeled 508). These port managers 508-1 to 508-J provide a bidirectional link (which can, for example, be a 10GBase-KR or 40GBase-KR like set forth in Institute of Electrical and Electronics Engineers (IEEE) standard 802.11ap on June 25, 2010 and IEEE standard 802.11ba on June 22, 2010) to PHYs (i.e., 206-1) through the media access controller or MAC 610. This MAC 610 is coupled to coupled to the shared memory 502 through a transmit pipeline (which generally comprises a transmit shared buffer interface 602 and a transmit first-in-first-out (FIFO) memory and controller 604) and a receive pipeline (which generally comprises a receive shared buffer interface 606 and a receive FIFO and controller 608). Additionally, as part of the search structure for switching circuit 402, port managers 508-1 to 508-J also include a packet FIFO and controller 612, head replacer 614, parser 616, and search interface that interact or communicate with the receive pipeline.

[0037] Looking first to the handling of received packets, packets are initially received by the MAC 610 of one of the port managers 508-1 to 508-J. Each received packet is temporarily stored in the receive FIFO and controller 608. For each packet, a packet descriptor 800 for each packet is created and stored in the receive shared buffer interface 606, while the packet is forwarded to the shared memory 502. These packet descriptors 800 (an example of which can be seen in FIG. 10) generally comprise a next packet descriptor pointer field 802 (which indicated the packet descriptor for the next or subsequent packet), a buffer descriptor pointer 804, a packet length 806, and a action set pointer 808 and provide an association with buffer descriptors 1100 used by the shared memory 502. The buffer descriptor 1100 (an example of which can be seen in FIG. 13) is generally the "address" for the packet in the shared memory 502 and generally comprises a buffer descriptor identifier field 1102, a linking information field 1104 (which is generally written by a direct memory

access controller in the interface 606), a buffer pointer field 1106 (which is generally a pointer to packet contents), a next pointer field 1108 (which is generally the next buffer descriptor), and a length field 1110 (which is generally the length of the buffer used in shared memory 502).

[0038] While the packet is being stored in shared memory 502, a lookup or search associated with the packet header is also performed. When each packet is passed to the receive FIFO and controller 608, the parser 616 (which is generally programmable) also receives the packet and extracts the packet header for each packet so as to construct a string of concatenated header fields. A lookup descriptor 1000 (an example of which is shown in FIG. 12) can then be formed for each packet and stored in the search interface 618. The lookup descriptor 1000 generally comprises a packet descriptor pointer field 1002 (which generally points to the associated packet descriptor 800), a buffer descriptor pointer field 1004 (which generally points to an associated buffer descriptor 1100), match fields 1008 (which is generally the concatenated header fields from parser 616), and an action set 902 (which is generally the set of actions to be performed on the packet). The action sets 902-1 to 902-T (as shown in the example of FIG. 11) for the packets are also generally stored in the action table 510 and are updated by the search engine 506.

[0039] Based on the lookup descriptor 1000 for each packet, the search engine 506 is able to perform a search to determine the appropriate actions to be taken. To do this, the search engine 506 uses to the primary table 406 for a "primary" binary entry search and the secondary table 408 for a "secondary" ternary entry search. Usually, a "primary" search (which is usually less "power hungry" than the "secondary" path) is followed by a "secondary" search, if the "primary" search is unsuccessful. Thus, the primary table can be thought of as a filter that reduces power consumption by limiting the use of the secondary table. Typically, ternary rules can be stored in secondary table 408-A, and the dynamic memory 410 can store binary versions of the ternary rules that are observed in actual packets. The location of dynamic memory 410 where a binary entry is stored can be computed by performing a hash function on the binary entry. This is driven by the insight that new flows are initiated much less frequently than the arrival of individual packets for each flow. Hence, flow set-up within a hash table can be done at order-of-magnitude slower pace.

[0040] With the "primary" path, a search on the primary table 406 for a binary rule is performed using a hash logic 704, where the dynamic memory 410 stores the binary rules together with a stash 412. The purpose of stash 412 is to store collided entries when multiple entries accidentally produce the identical hash function output. One or more memory arrays or banks (such as static random access memories (SRAMs) 414-1 to 414-I or embedded dynamic random access memory (eDRAM) shown in FIGS. 8 and 9) can comprise the dynamic memory 410, and the stash 412 is generally comprised of a CAM 416. Typically, the controller 702 applies a data word to the hash logic 704 based on a lookup descriptor (i.e., 1000). The hash logic 704 applies a hash function (which may be keyed for security purposes) to the match fields (1008) of the lookup descriptor so that a binary search of the dynamic memory 406 can be performed. The hash logic 704 generally implements a multi-level hash table with multiple subtables 414-1 to 414-I with independent hash functions. Typically, the dynamic memory 406 stores tables having entries (which can be referred to as rules) that associate match fields with a priority. Matches can then be returned for each substable. Additionally, a list of example match fields can be seen in Table 1 below.

Table 1

Field	Width	When Applicable
Ingress Port	32	All packets
Metadata	64	
Ethernet Source Addr.	48	All packets on enable ports
Ethernet Dest. Addr.	48	All packets on enable ports
Ethernet type	16	All packets on enable ports
virtual local area network (VLAN) identifier	12	All packets with VLAN tags
VLAN priority	3	All packets with VLAN tags

Table 1

Field	Width	When Applicable
Multiprotocol Label Switching (MPLS) label	20	All packets with MPLS tags
MPLS traffic class	3	All packets with MPLS tags
Internet Protocol version 4 (IPv4) Source Addr.	32	All IPv4 and Address Resolution Protocol (ARP) packets
IPv4 Dest. Addr.	32	All IPv4 and ARP packets
IPv4 protocol / Address Resolution ARP opcode	8	All IPv4, IPv4 over Ethernet, and ARP packets
IPv4 Type of Service (ToS) bits	6	All IPv4 packets
Transport Source Port / Internet Control Message Protocol (ICMP) type	16	All Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), and ICMP packets
Transport Dest. Port / ICMP code	16	All TCP, UDP, SCTP, and ICMP packets

[0041] As mentioned above, the hash logic 704 may be keyed for security purposes. As an example, the hash logic 704 generally implements a multi-level hash table with

subtables T_1 to T_d with hash functions h_1 to h_d. A keyed hash on a binary string x with subtable T_w can, for example, be:

(1) $h_w(x)=((a_w x+b_w) \bmod P) \bmod N_w$

where P is a large prime number, a_w and b_w (which are each less than P) for the key pair, and N_w is maximum number of entries in the subtable T_w. Parallel searches for the subtables T_1 to T_d can then be performed.

[0042] As part of maintaining, the primary table 406, the hash logic 704 can also add binary strings or rules to the primary table 406. To add a binary table entry or a binary string x (for example) to the primary table 406, hash function $h_w(x)$ is calculated for every subtable w, and an attempt is made to place string x into location $h_w(x)$ in any of the subtables w, when that location is vacant. If no location $h_w(x)$ is vacant, string x is inserted into the stash 412. Alternatively, when hash logic 704 is implemented as a cuckoo hash, string x can be inserted into $h_1(x)$, and a string y that occupied $h_1(x)$ is rehashed as string y into one of the vacant locations $h_w(y)$ in any of the subtables w. If all locations $h_w(y)$ are occupied, then string y is inserted into the stash 412. In effect, the hash logic 704 adds binary entries into the primary table 406 and can lookup binary entries from the primary table 406.

[0043] When, for example, no rule matching the hashed data word associated with the header for a packet can be found (which can be referred to as a match failure) during a "primary" search, further processing is performed. When a match failure occurs, the associated lookup descriptor (i.e., 1000) is stored in the packed descriptor queue 706, which generally operates as a temporary memory because of the speed difference between lookups in the primary table 406 and secondary table 408. In case there is no speed difference between lookups in primary and secondary tables (but only a power difference), the queue 706 can be omitted.

[0044] Then, a ternary search of the secondary table 408 (which can be formed of TCAM banks 418-1 to 418-R in the secondary table 408-A of FIG. 8 or can be formed of SRAM banks 420-1 to 420-R in the secondary table 408-B of FIG. 9) is performed using the match fields (1008) of the lookup descriptor (i.e., 1000). Typically, the secondary table 408 is formed of several banks of memory (as shown in FIGS. 8 and 9) that can each contains a ternary rule table. Replicated copies of the lookup descriptor (which did not yield a match in the "primary" path) can then be used to search the ternary rule tables substantially at the

same time. Other search methods may also be employed. A match can then yield instructions for the action table 510. Additionally, for each match found within the secondary table 408, a new binary search rule can be created in the dynamic memory 406 for future use. More specifically, binary versions of the ternary rules that are observed in actual packets are inserted in the primary table 406. In the event that no match is found, the packet header associated with the match failure can be encapsulated and sent to the processor 402 or network controller 401 for further processing; alternately, the packet is dropped.

[0045] Usually, with match failures in the "secondary" path, a modification to the tables of the "secondary" path may be useful. In many cases, when there is a match failure in the "secondary" path, an adequate rule may be missing from the secondary table 408, so the processor 402 or network controller 401 can "insert" a new rule. Usually, the new rules are added to the banks of the secondary table 408 in a round-robin fashion to achieve load balancing among across the secondary table 408. Additionally, rules in the secondary table 408 or in primary table 406 may be removed or evicted based on a "least recently used" measure or some other statistics.

[0046] Once the rules or actions associated with each packet's header have been resolved. The packet can be modified for further processing and/or routing. This is generally achieved by header replacer 614. Typically, the header replacer 614 modifies the packet descriptor 800 for each packet by associating the action set pointer 808 with the proper action set in the action table 510 using the packet FIFO and controller 612 and receive FIFO and controller 608.

[0047] With transmit packets, the handling in port managers 508-1 to 508-J is somewhat simpler compared to received packets. Usually, processing of the packets for routing has been completed prior to transmission. When the routing has been determined a destination port 208-1 to 208-P is usually packet. As a result, the appropriate port manager 508-1 to 508-J recalls packet information from the shared memory 502 using the transmit shared buffer interface 602, and this completed packet is temporarily stored in the transmit FIFO and controller 604. The MAC 610 can then distributed the packet to the appropriate PHY (i.e., 206-1).

[0048] Those skilled in the art to which the invention relates will appreciate that modifications may be made to the described example embodiments, and also that many other embodiments are possible, within the scope of the claimed invention.

CLAIMS

What is claimed is:

1. An apparatus comprising:

a lookup memory having a primary table and a secondary table, wherein the secondary table includes a first set of ternary rules, wherein the primary table includes a set of binary rules, and wherein each binary rule is a binary version of at least one ternary rule from a second set of ternary rules; and

a search engine coupled to the lookup memory, wherein the search engine includes:

a controller configured to receive data words; and

hash logic coupled to the lookup memory and to the controller, wherein the hash logic is configured to perform a binary search of the primary table to determine whether each data word matches at least one of the binary rules, and wherein, if there is a failed match by hash logic and primary table, the search engine is configured to perform a ternary search of the secondary table to determine whether the data word associated with the failed match matches at least one of the ternary rules from the first set of ternary rules.

2. The apparatus of Claim 1, wherein the primary table further comprises a dynamic memory; and a stash memory.

3. The apparatus of Claim 2, wherein the stash memory is a content-addressable memory (CAM).

4. The apparatus of Claim 3, wherein the dynamic memory is a static random access memory (SRAM).

5. The apparatus of Claim 4, wherein the secondary table further comprises a Ternary CAM (TCAM).

6. The apparatus of Claim 4, wherein the apparatus further comprises:

a shared memory;

a plurality of port managers, wherein each port manager includes communication circuitry that is configured to receive input data packets and that is coupled to the shared memory and the search engine; and

a parser coupled to the communication circuitry, wherein the parser is configured to parse each input data packet and extract its header, and wherein each data word is associated with at least one header.

7. The apparatus of Claim 6, wherein the apparatus further comprises an action table that is in communication with the search engine.

8. The apparatus of Claim 7, wherein the communication circuitry further comprises:

a media access controller (MAC) that is coupled to the parser;

a transmit pipeline that is coupled between the shared memory and the MAC;

a receive pipeline that is coupled between the shared memory and the MAC; and

a search interface that is coupled between the parser and the search engine.

9. The apparatus of Claim 8, wherein the hash logic applies a keyed hash function to each data word.

10. A method comprising:

receiving a plurality of packets, wherein each packet has a body and a header;

parsing each packet to extract its header;

applying a hash function to each header;

comparing each hashed header with a plurality of binary rules stored within a primary table, wherein each binary rule is a binary version of at least one ternary rule from a first set of ternary rules; and

for each match failure with the plurality of rules, searching a secondary table using the header associated with each match failure, wherein the secondary table includes a second set of ternary rules.

11. The apparatus of Claim 10, wherein the primary table further comprises an SRAM and
a CAM.

12. The method of Claim 11, wherein the secondary table is a TCAM.

13. The method of Claim 12, wherein the step of searching the secondary table further comprises simultaneously searching a plurality of banks within the TCAM.

14. The method of Claim 11, wherein the method further comprises generating a new rule and a new action for each match failure; and storing the new rule and new action in the SRAM.

15. The method of Claim 14, wherein the hash function is a keyed hash function.

FIG. 1
(PRIOR ART)

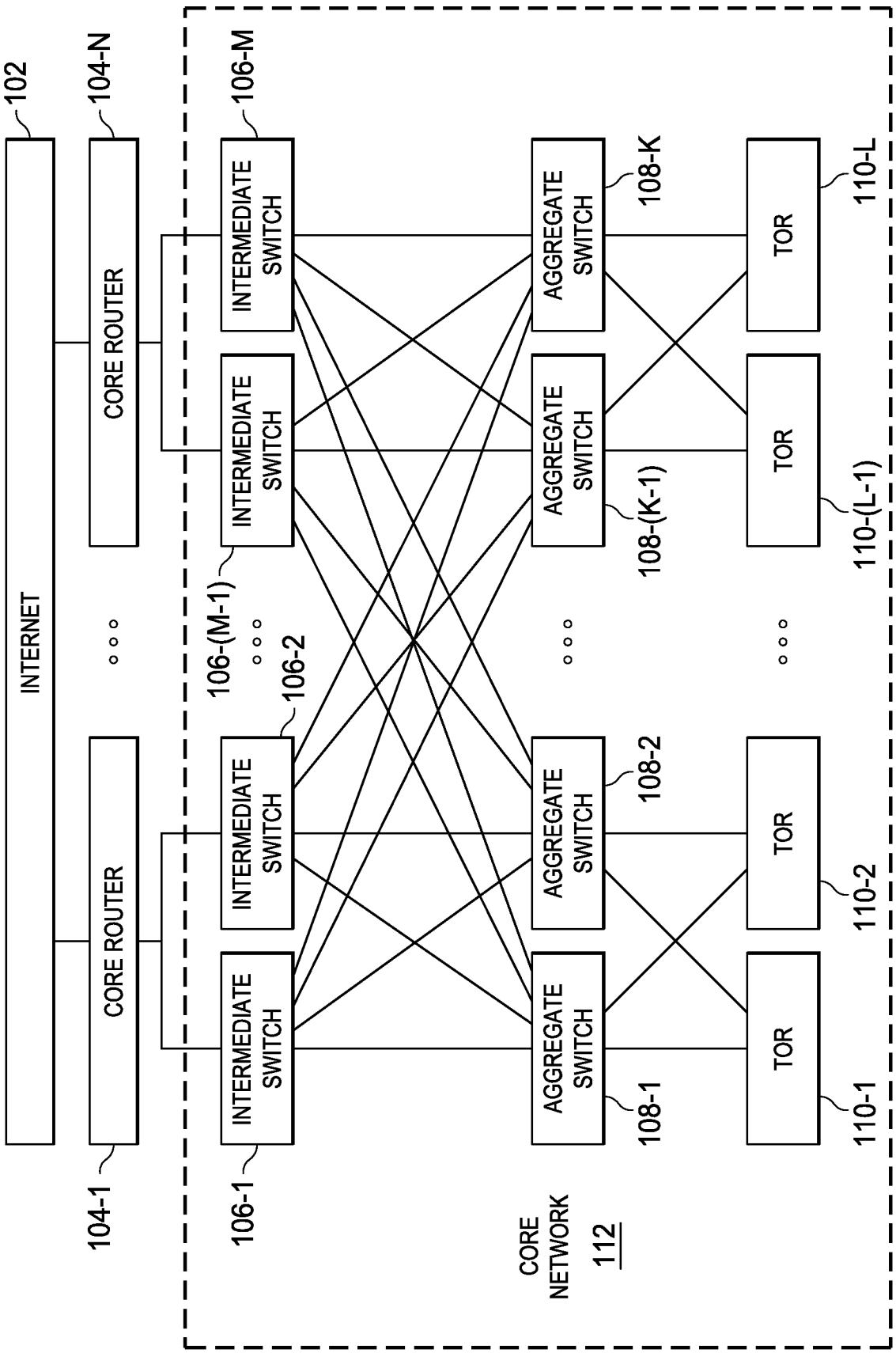


FIG. 2
(PRIOR ART)

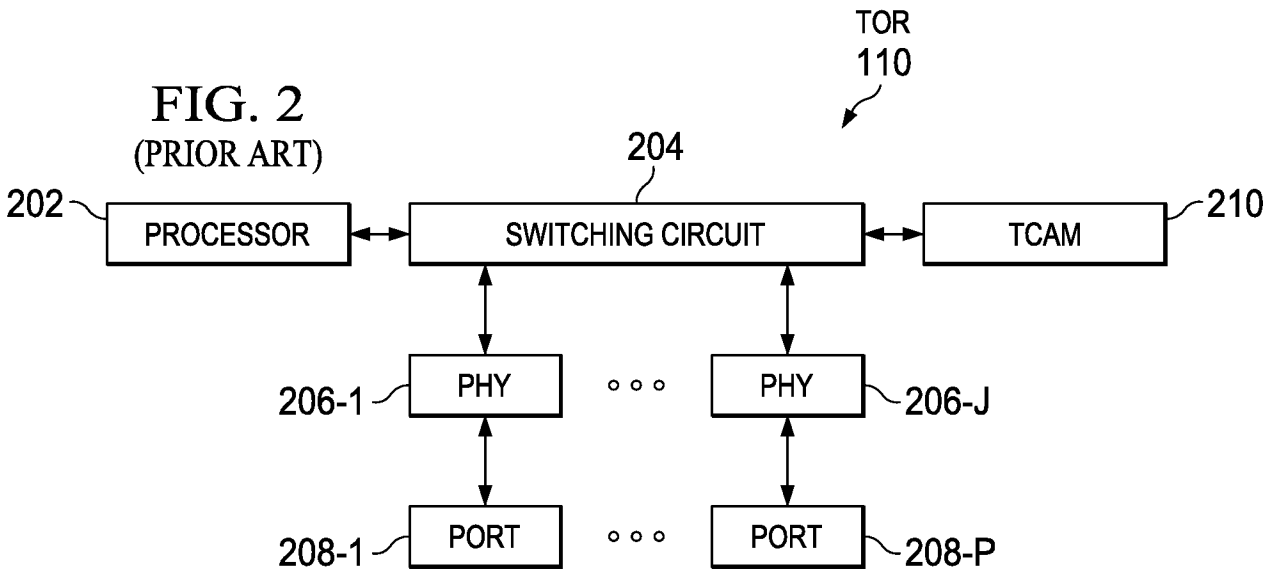


FIG. 3
(PRIOR ART)

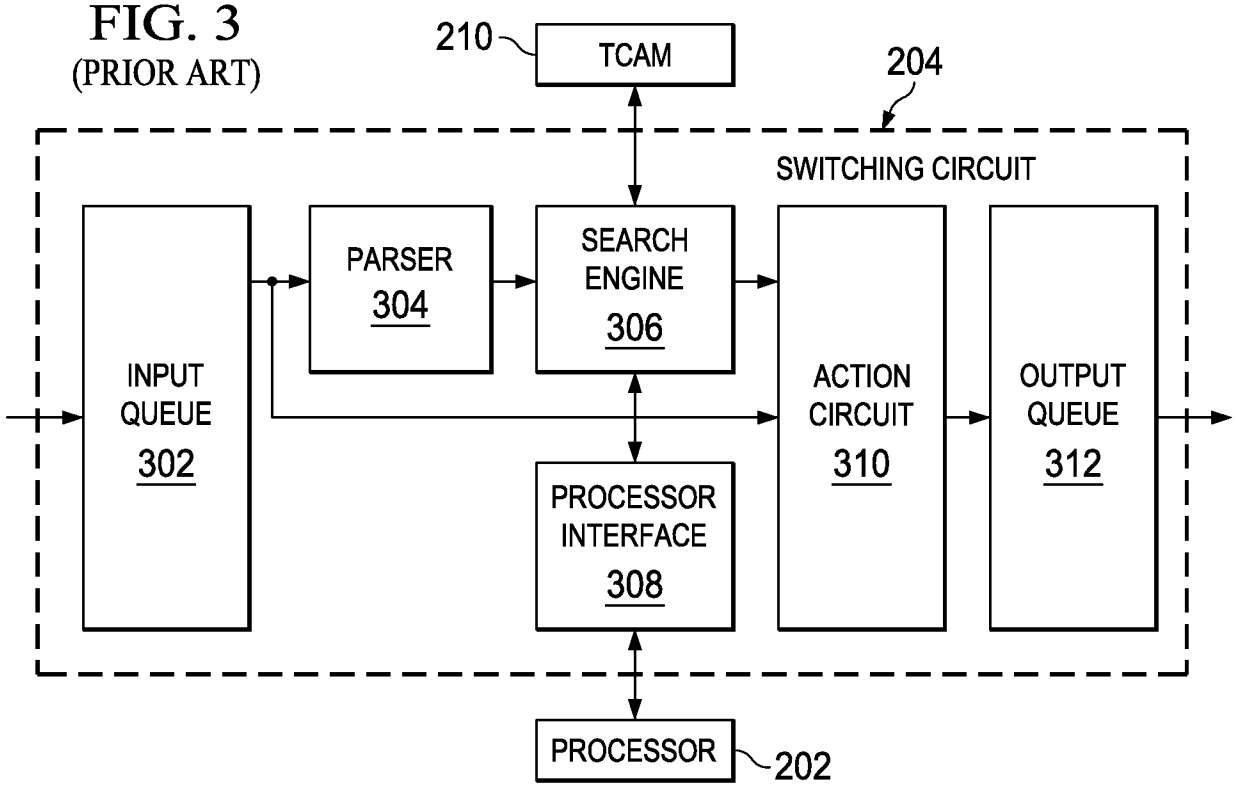


FIG. 4

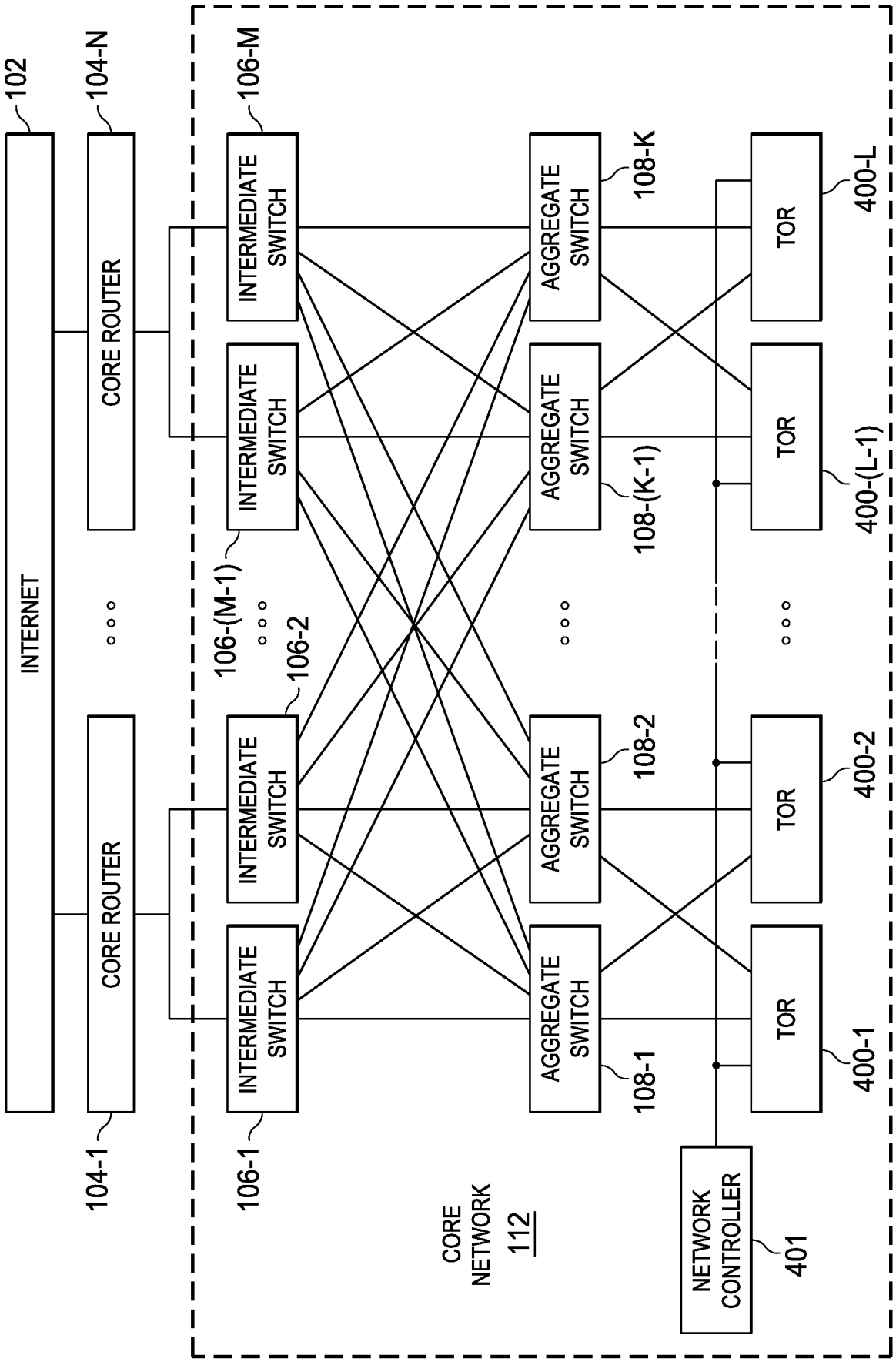

INTERNET
102
CORE ROUTER
104-N
CORE ROUTER
104-1
INTERMEDIATE SWITCH
106-M
INTERMEDIATE SWITCH
106-(M-1)
INTERMEDIATE SWITCH
106-2
INTERMEDIATE SWITCH
106-1
AGGREGATE SWITCH
108-K
AGGREGATE SWITCH
108-(K-1)
AGGREGATE SWITCH
108-2
AGGREGATE SWITCH
108-1
TOR
400-L
TOR
400-(L-1)
TOR
400-2
TOR
400-1
NETWORK CONTROLLER
401
CORE NETWORK
112

FIG. 5

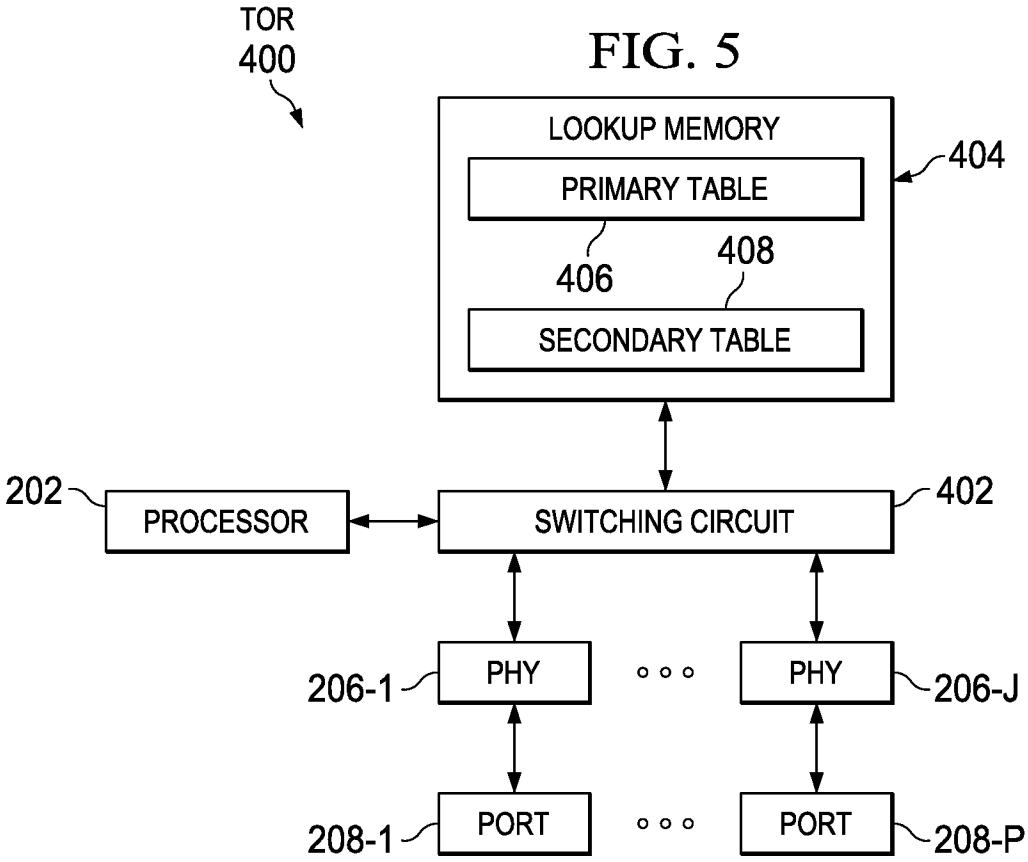
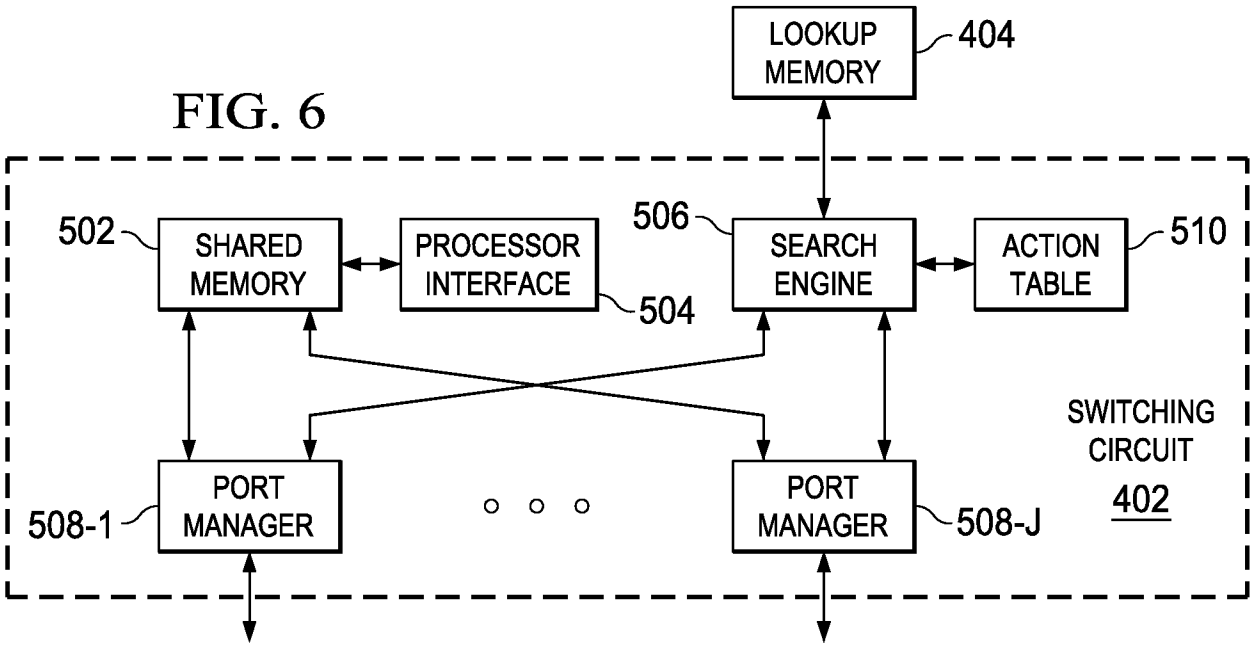

TOR
400
LOOKUP MEMORY
404
PRIMARY TABLE
406
408
SECONDARY TABLE
202
PROCESSOR
SWITCHING CIRCUIT
402
206-1
PHY
PHY
206-J
208-1
PORT
PORT
208-P

FIG. 6


LOOKUP MEMORY
404
502
SHARED MEMORY
PROCESSOR INTERFACE
504
506
SEARCH ENGINE
ACTION TABLE
510
SWITCHING CIRCUIT
402
508-1
PORT MANAGER
PORT MANAGER
508-J

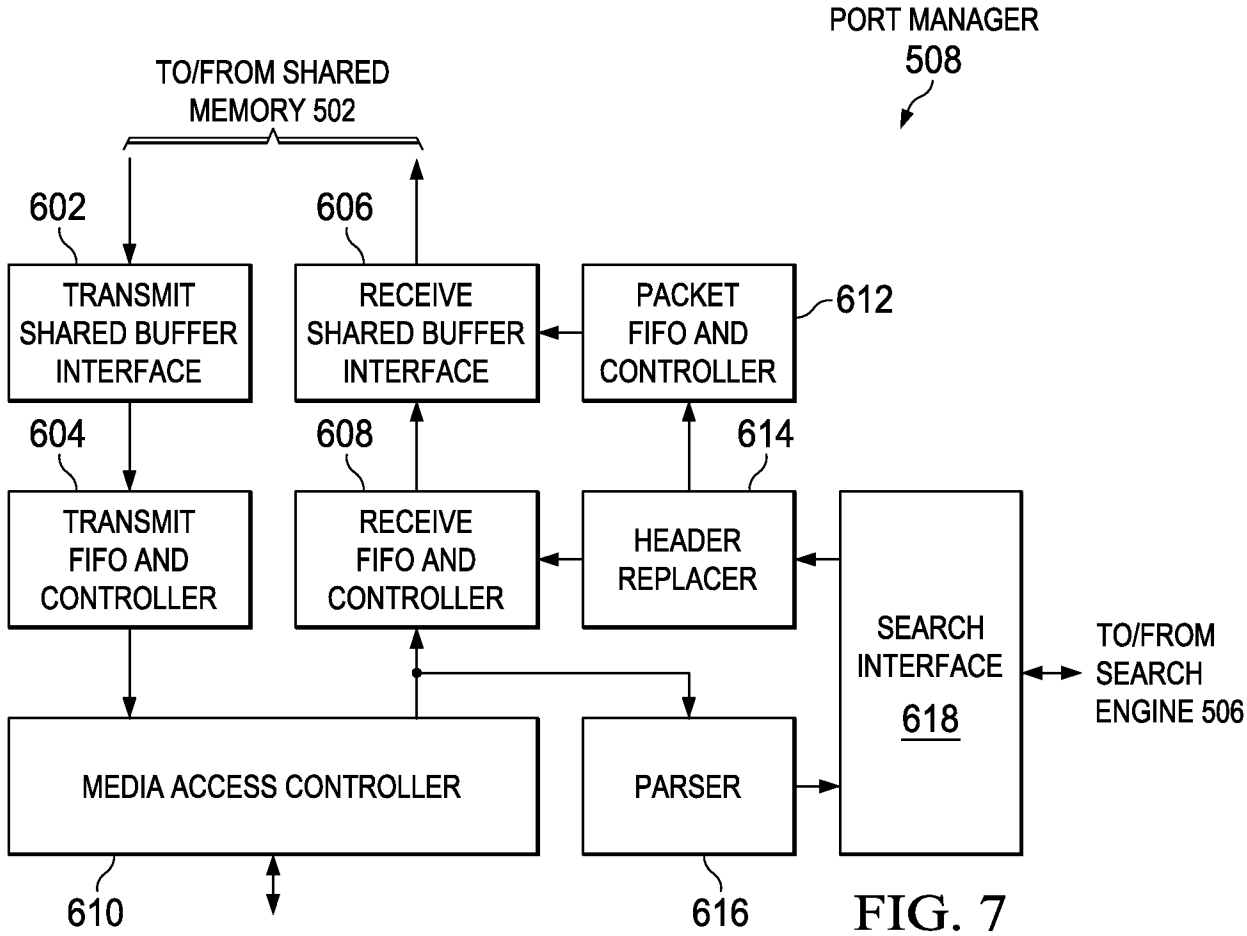

PORT MANAGER
508
TO/FROM SHARED MEMORY 502
602
606
TRANSMIT SHARED BUFFER INTERFACE
RECEIVE SHARED BUFFER INTERFACE
PACKET FIFO AND CONTROLLER
612
604
608
614
TRANSMIT FIFO AND CONTROLLER
RECEIVE FIFO AND CONTROLLER
HEADER REPLACER
SEARCH INTERFACE
618
TO/FROM SEARCH ENGINE 506
MEDIA ACCESS CONTROLLER
PARSER
610
616

FIG. 7

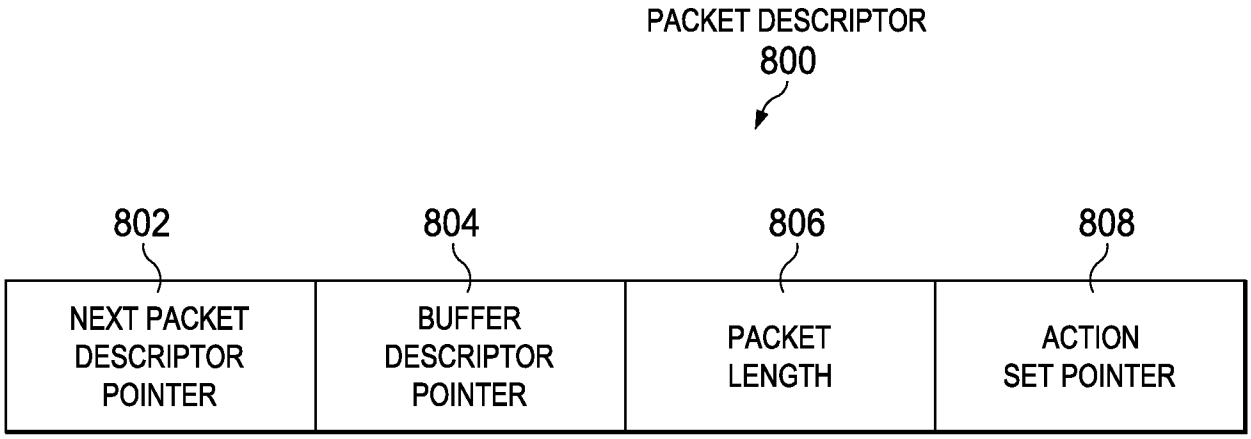
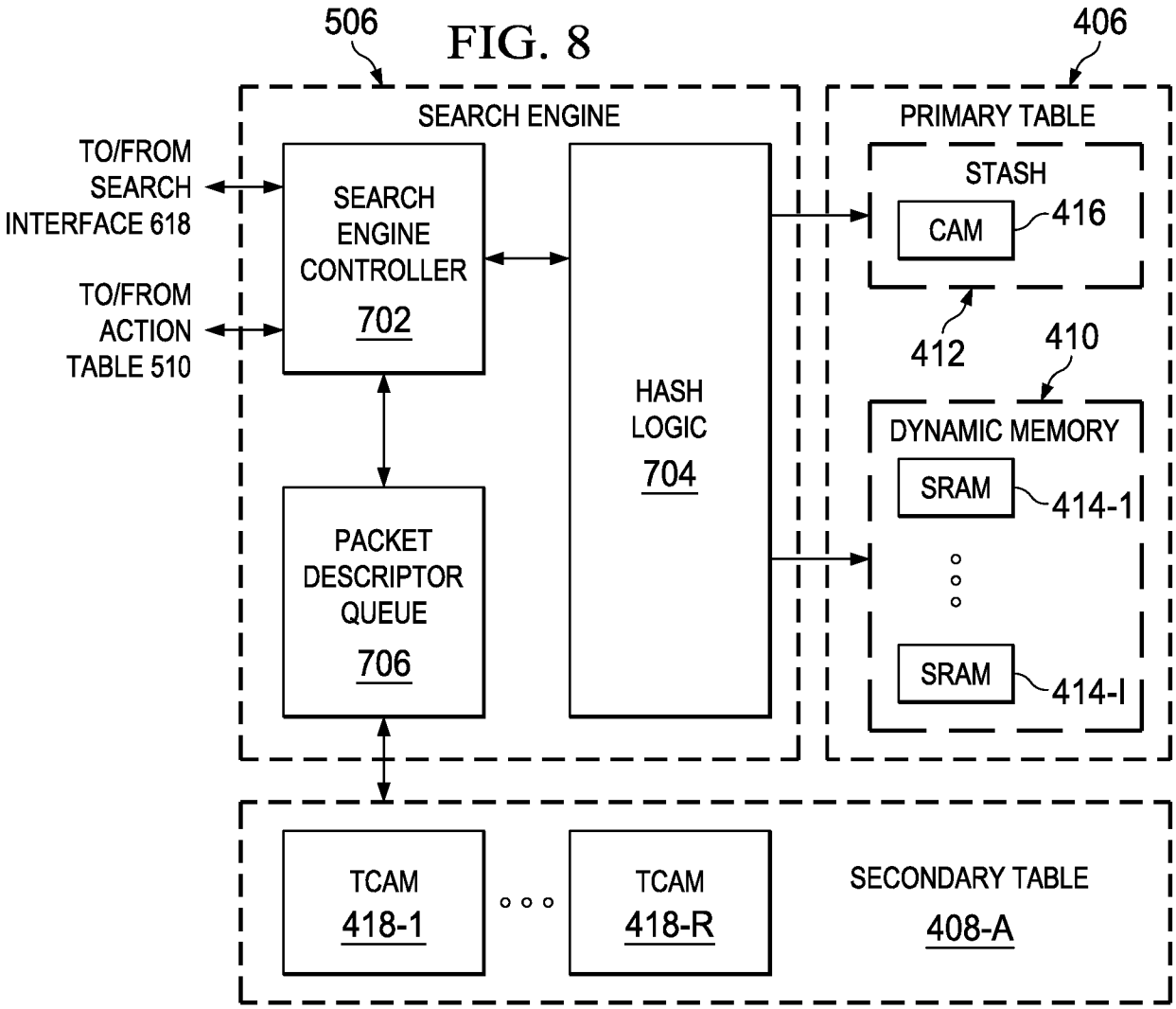
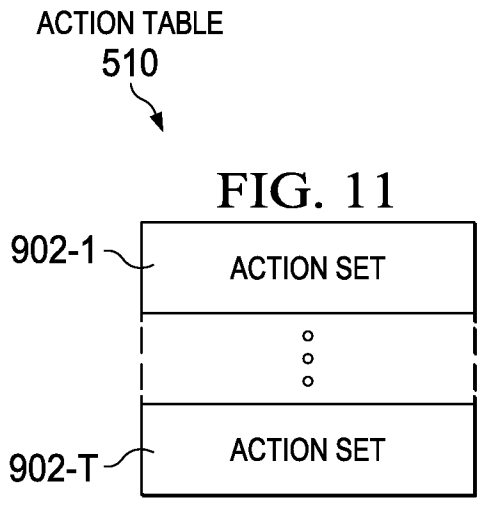

PACKET DESCRIPTOR
800
802
804
806
808
NEXT PACKET DESCRIPTOR POINTER
BUFFER DESCRIPTOR POINTER
PACKET LENGTH
ACTION SET POINTER

FIG. 10

FIG. 8


506
406
SEARCH ENGINE
PRIMARY TABLE
TO/FROM SEARCH INTERFACE 618
TO/FROM ACTION TABLE 510
SEARCH ENGINE CONTROLLER 702
HASH LOGIC 704
PACKET DESCRIPTOR QUEUE 706
STASH
CAM
416
412
410
DYNAMIC MEMORY
SRAM
414-1
SRAM
414-I
TCAM 418-1
TCAM 418-R
SECONDARY TABLE 408-A


ACTION TABLE 510
FIG. 11
902-1
ACTION SET
902-T
ACTION SET

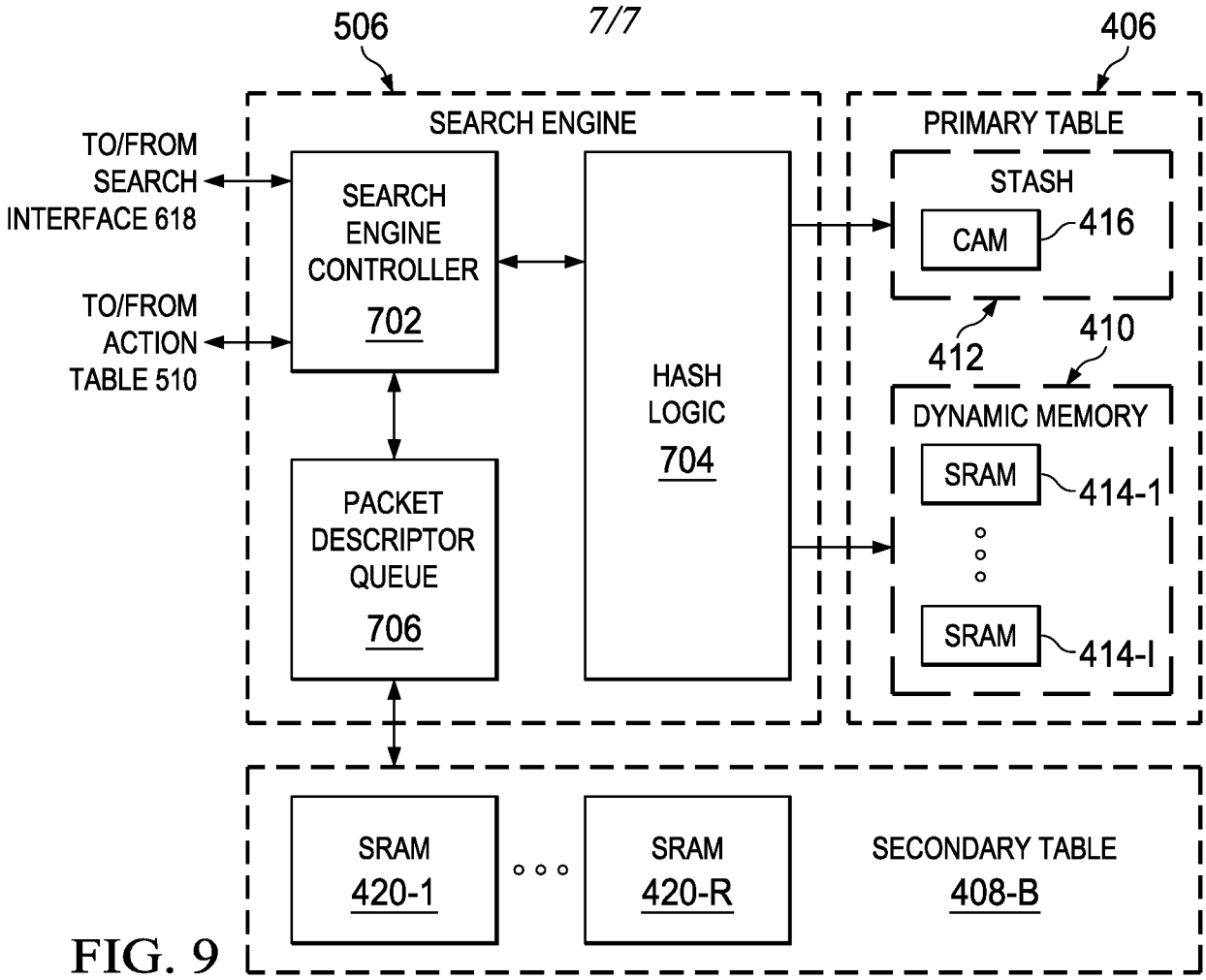


FIG. 9

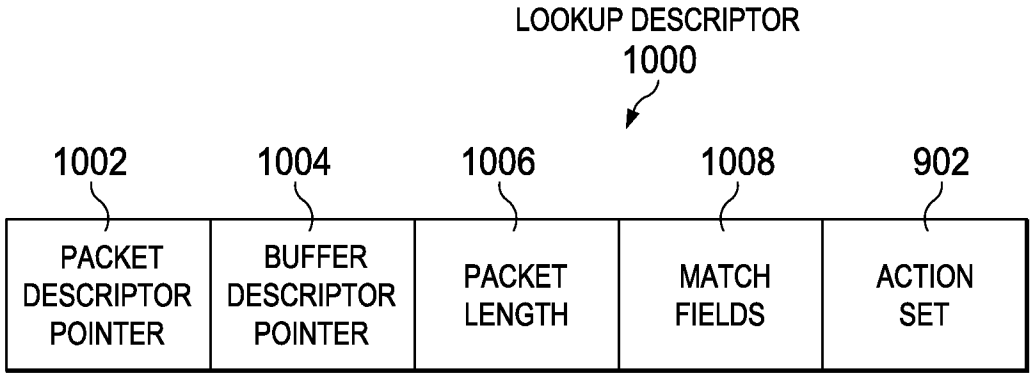


FIG. 12

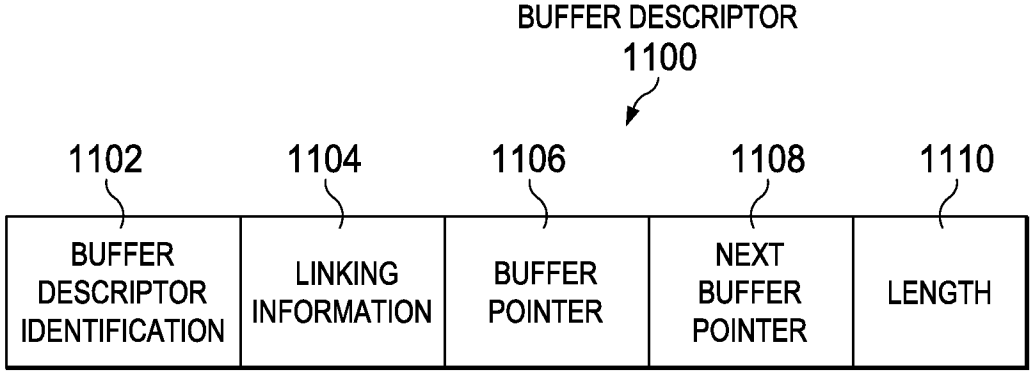


FIG. 13