



(19) **United States**

(12) **Patent Application Publication**

**Hegde et al.**

(10) **Pub. No.: US 2004/0131079 A1**

(43) **Pub. Date:**

**Jul. 8, 2004**

(54) **APPARATUS AND METHOD FOR CONFIGURING DATA PLANE BEHAVIOR ON NETWORK FORWARDING ELEMENTS**

(52) **U.S. Cl. .... 370/466**

(76) **Inventors: Shriharsha S. Hegde, Beaverton, OR (US); Russell J. Fenger, Beaverton, OR (US); Amol Kulkarni, Beaverton, OR (US); Hsin-Yuo Liu, Beaverton, OR (US); Hormuzd M. Khosravi, Hillsboro, OR (US); Manasi Deval, Beaverton, OR (US)**

(57) **ABSTRACT**

A method and apparatus for configuring data plane behavior on network forwarding elements are described. In one embodiment, the method includes receiving, within a network element control plane, protocol configuration information extracted from a protocol application utilizing a network protocol application programming interface (API). Once the protocol configuration information is received, the protocol configuration information is processed using a control interface corresponding to the network protocol implemented by the protocol application. Once the protocol configuration information is processed, the control interface programs one or more data plane forwarding elements of the network element according to protocol configuration information. Accordingly, by providing similar control interfaces for multiple, network protocols, inter-operability between components from multiple vendors is enabled.

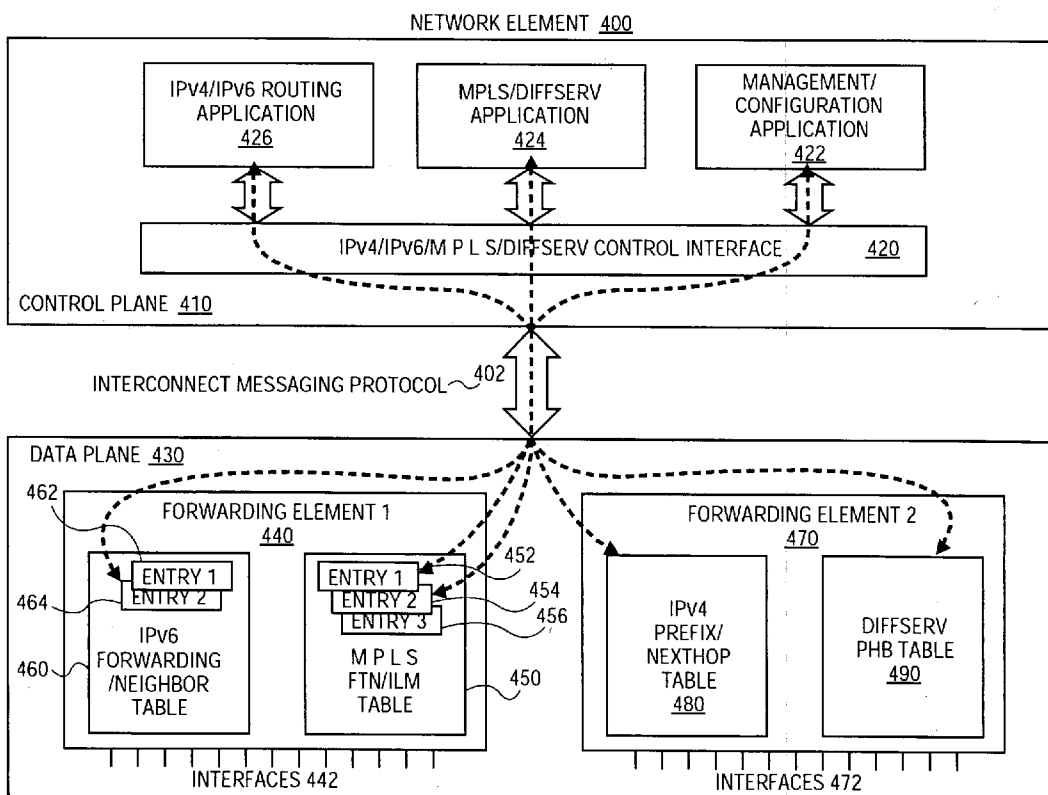
Correspondence Address:  
**BLAKELY SOKOLOFF TAYLOR & ZAFMAN**  
**12400 WILSHIRE BOULEVARD, SEVENTH FLOOR**  
**LOS ANGELES, CA 90025 (US)**

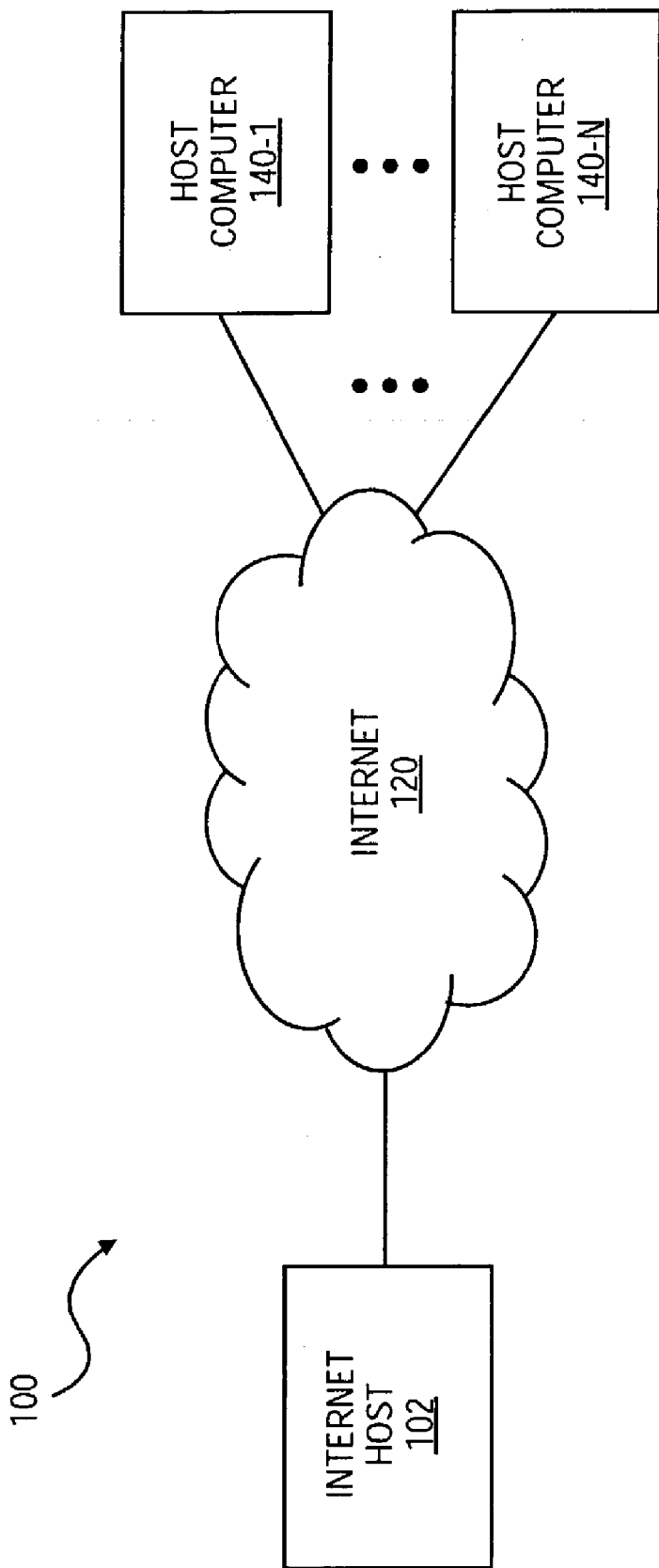
(21) **Appl. No.: 10/338,291**

(22) **Filed: Jan. 7, 2003**

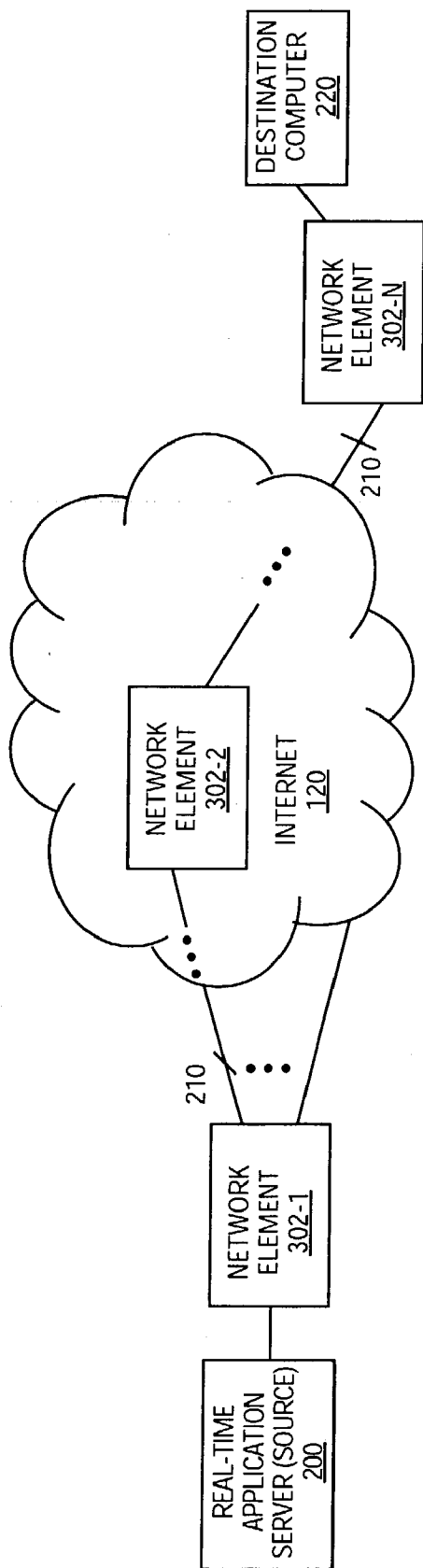
**Publication Classification**

(51) **Int. Cl.<sup>7</sup> ..... H04J 3/16**



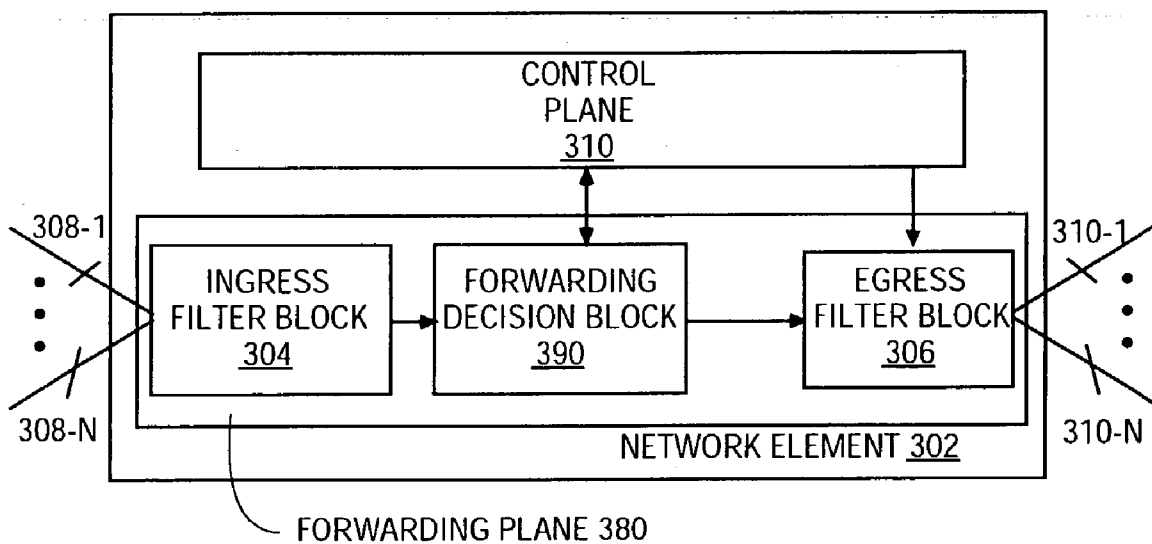


**FIG. 1**



**FIG. 2**

CONVENTIONAL NETWORK 300



**FIG. 3**  
(PRIOR ART)

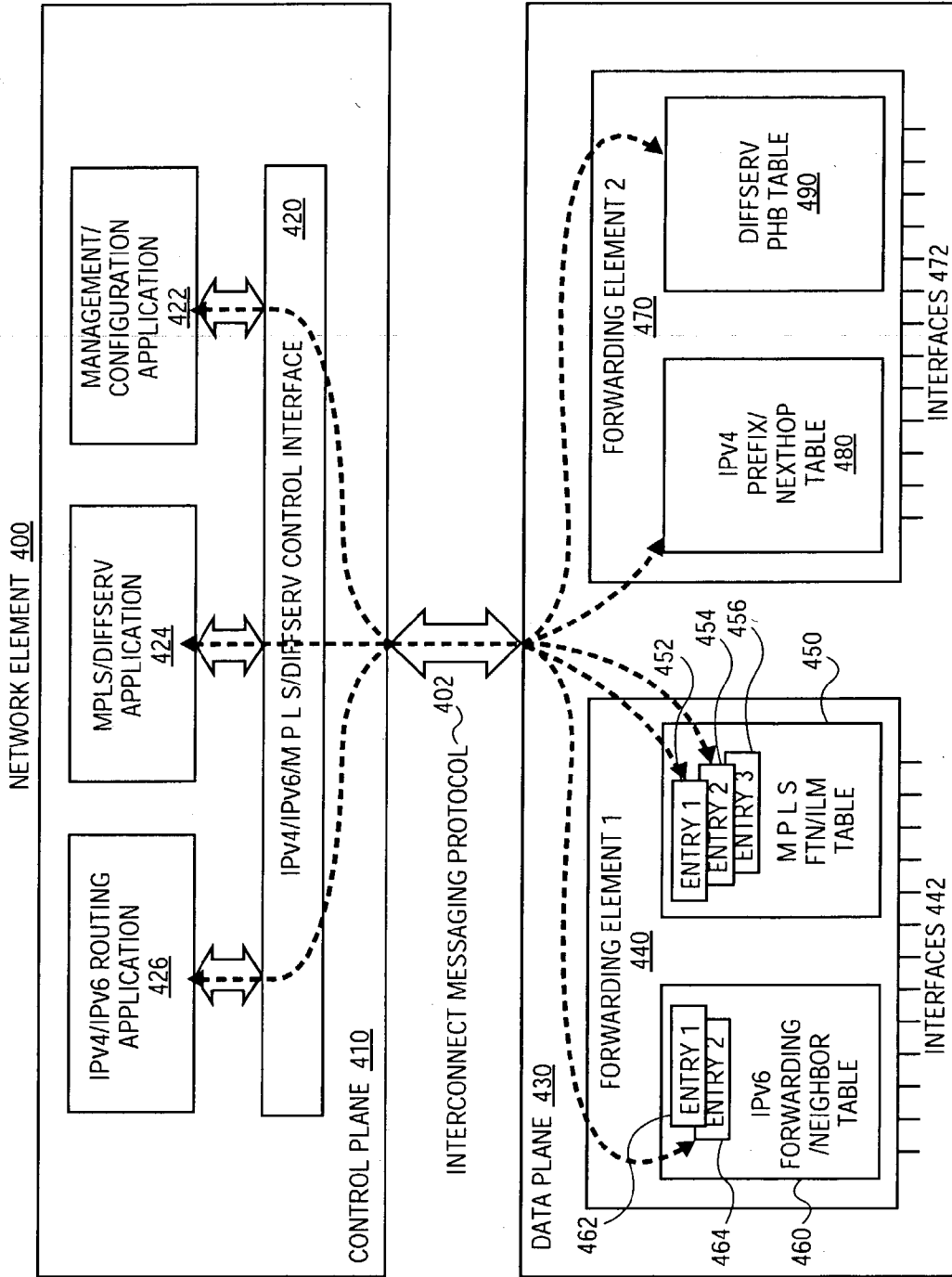
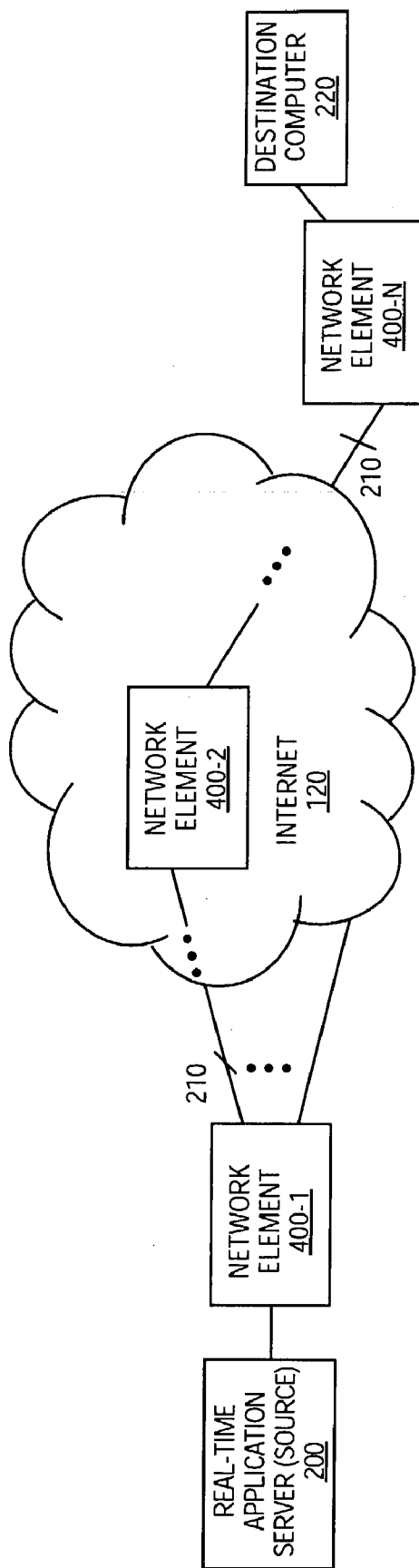
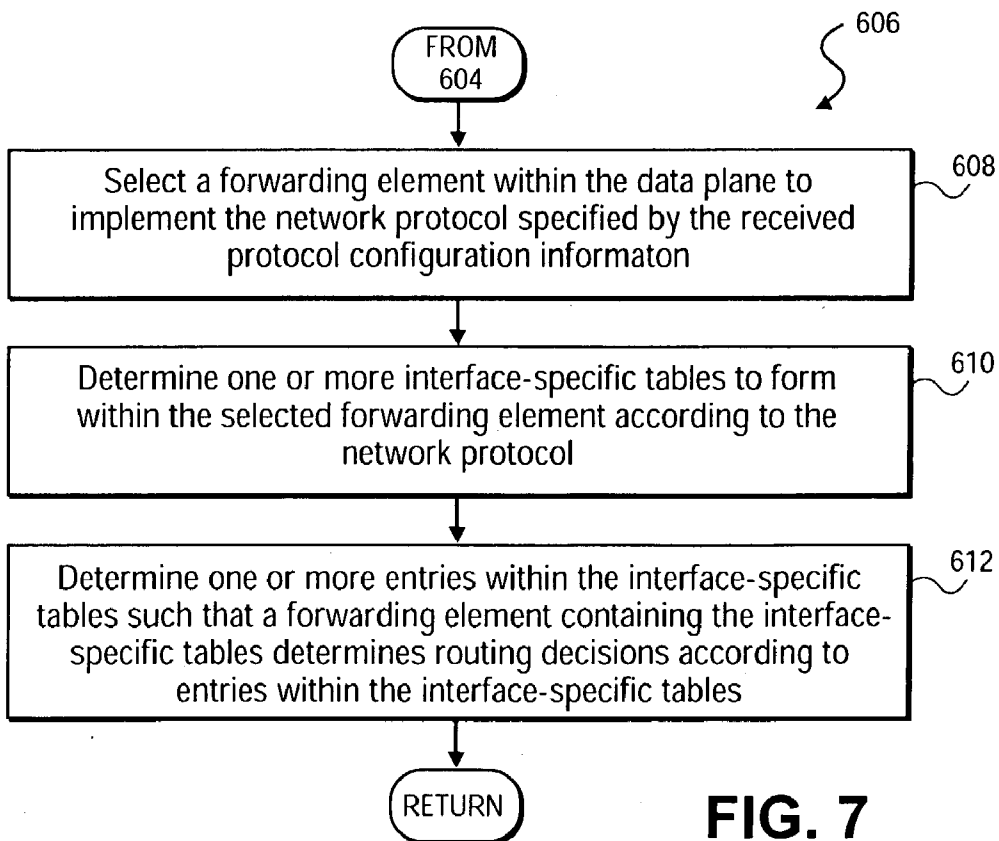
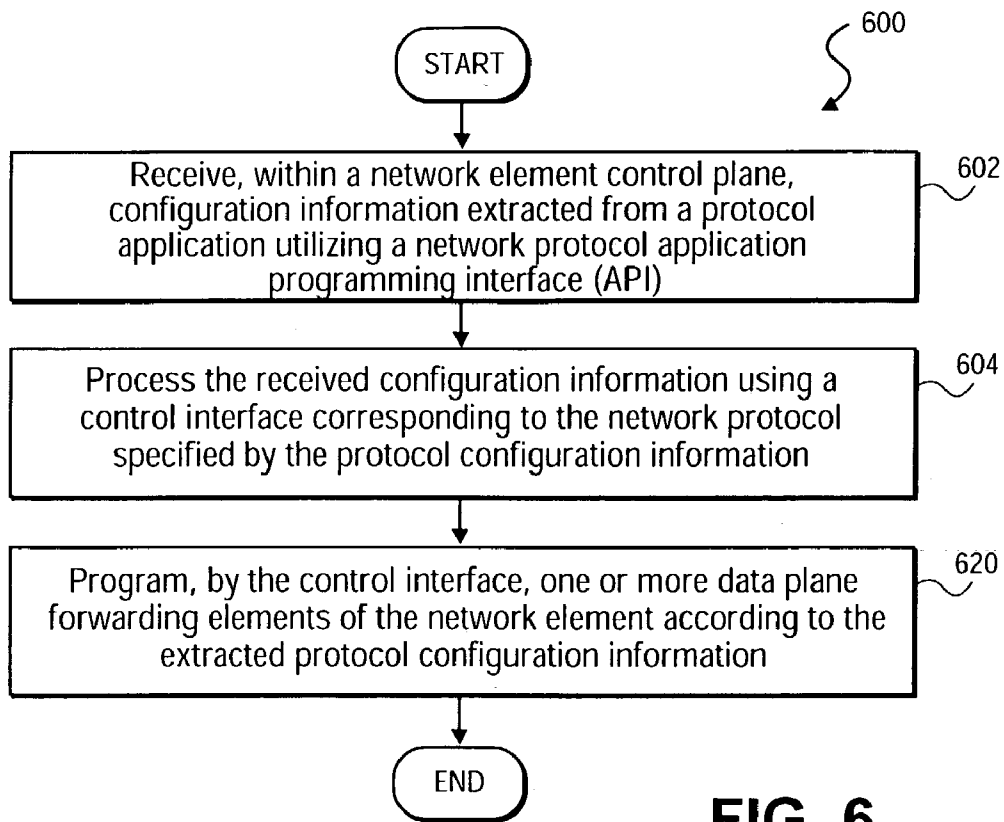


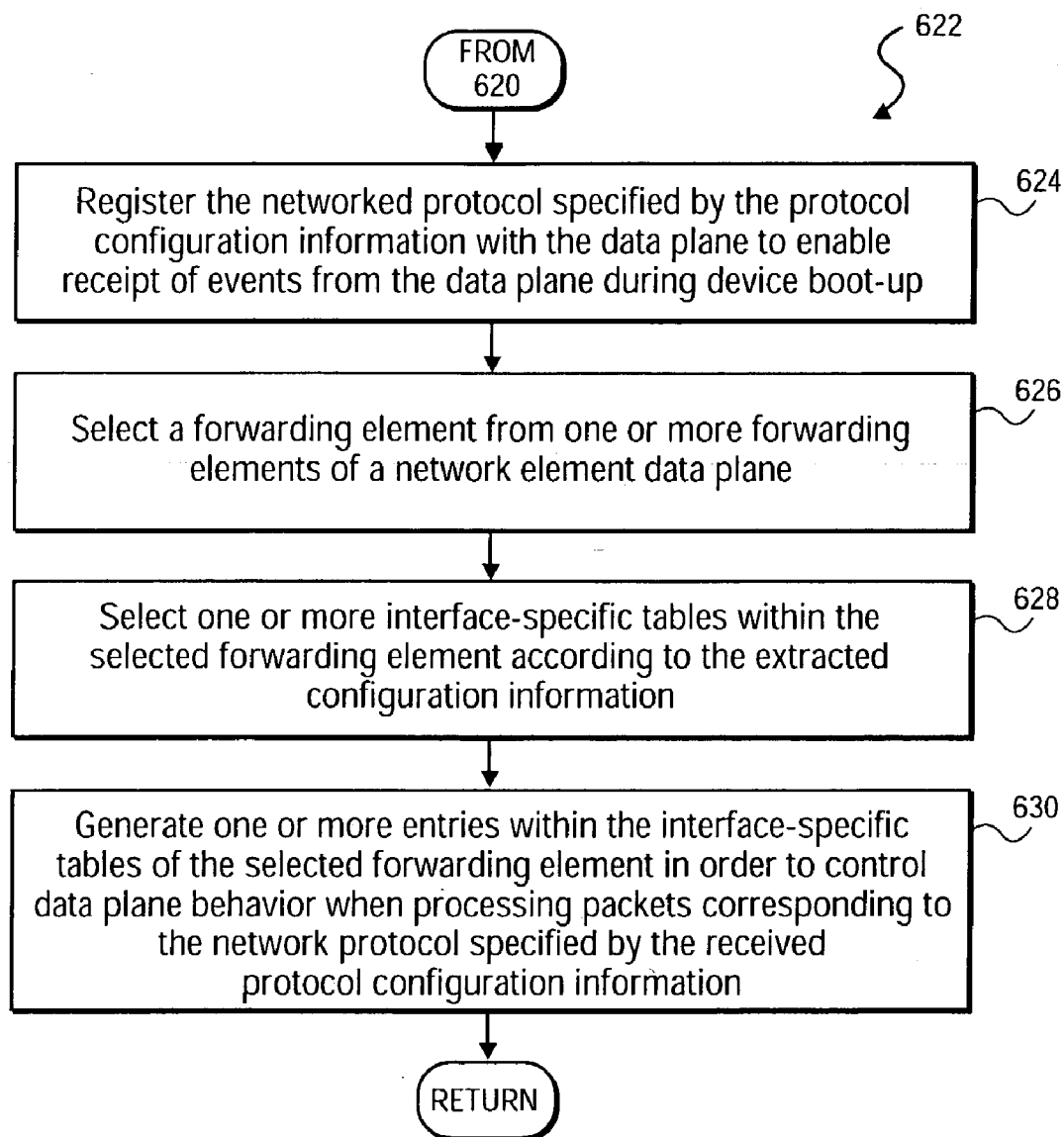
FIG. 4



**FIG. 5**

NETWORK 500





**FIG. 8**

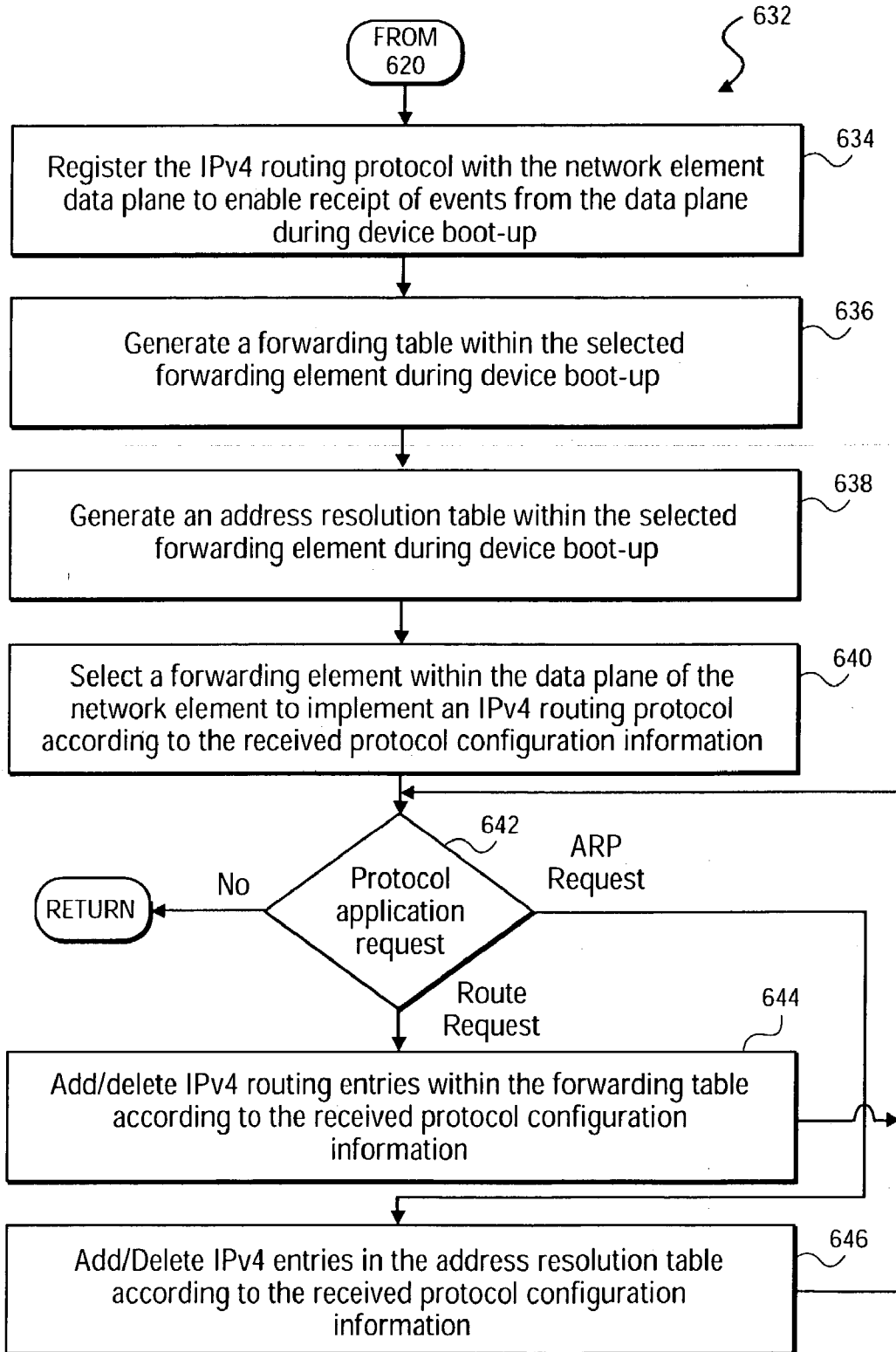


FIG. 9

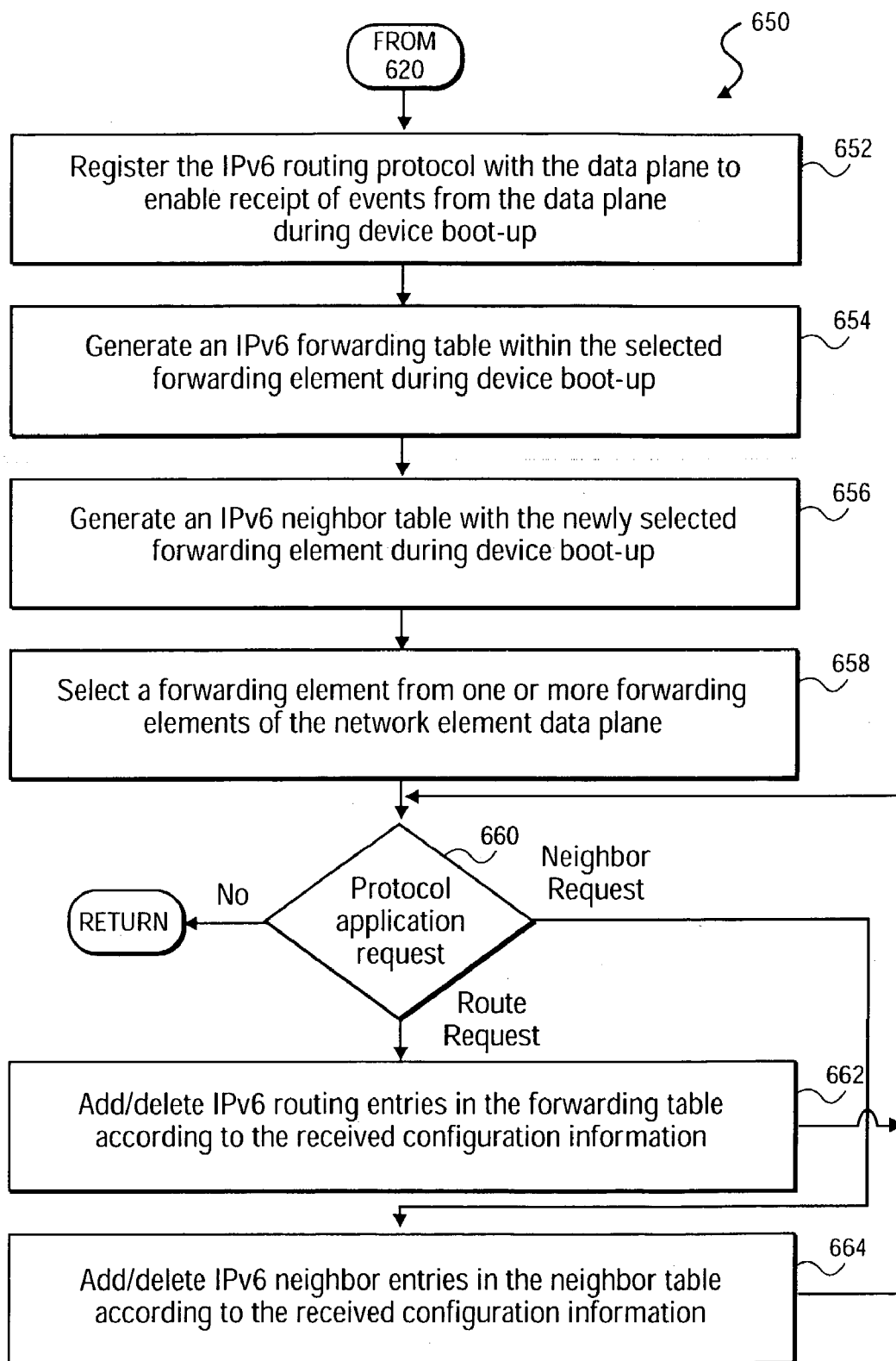


FIG. 10

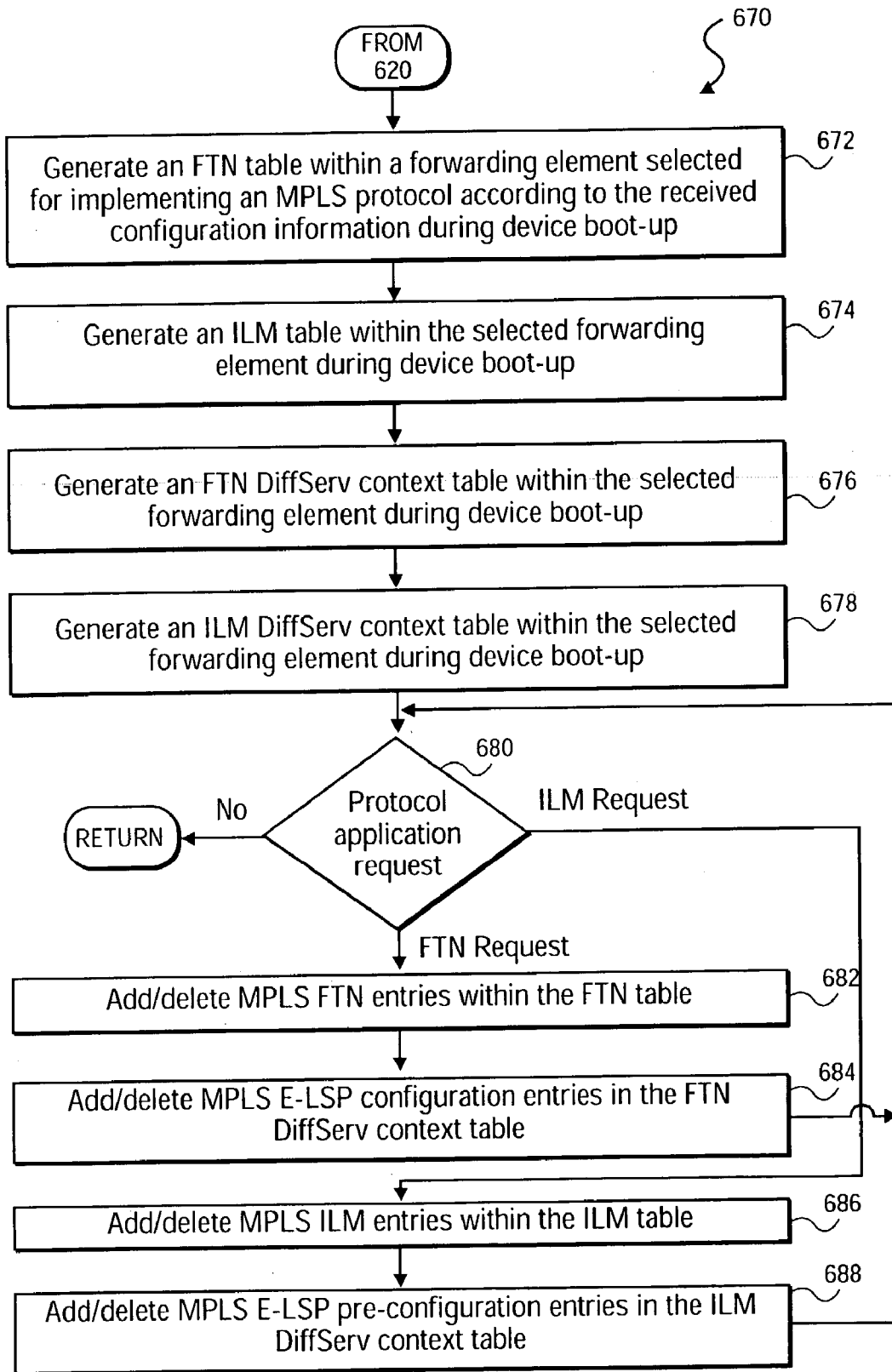


FIG. 11

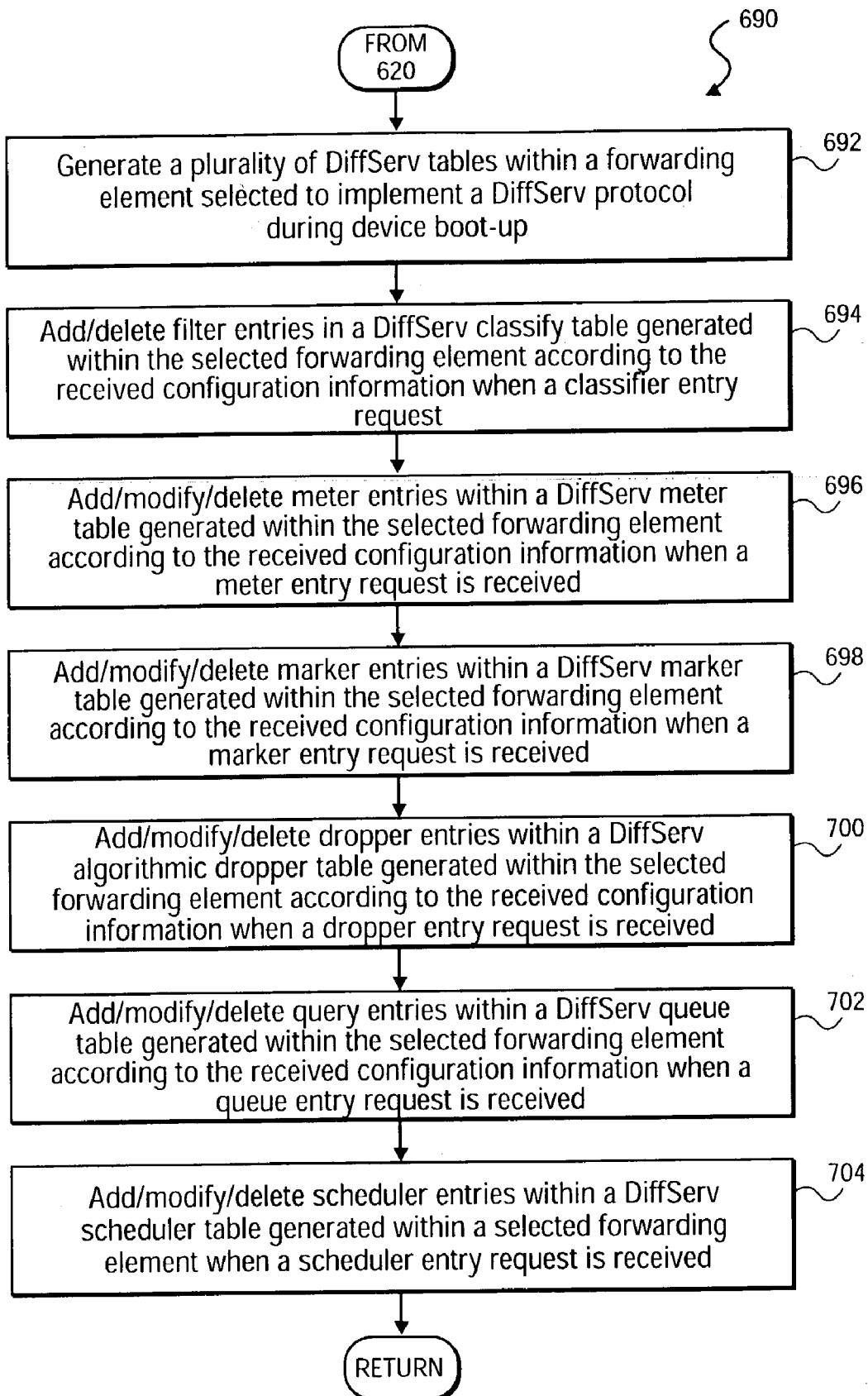


FIG. 12

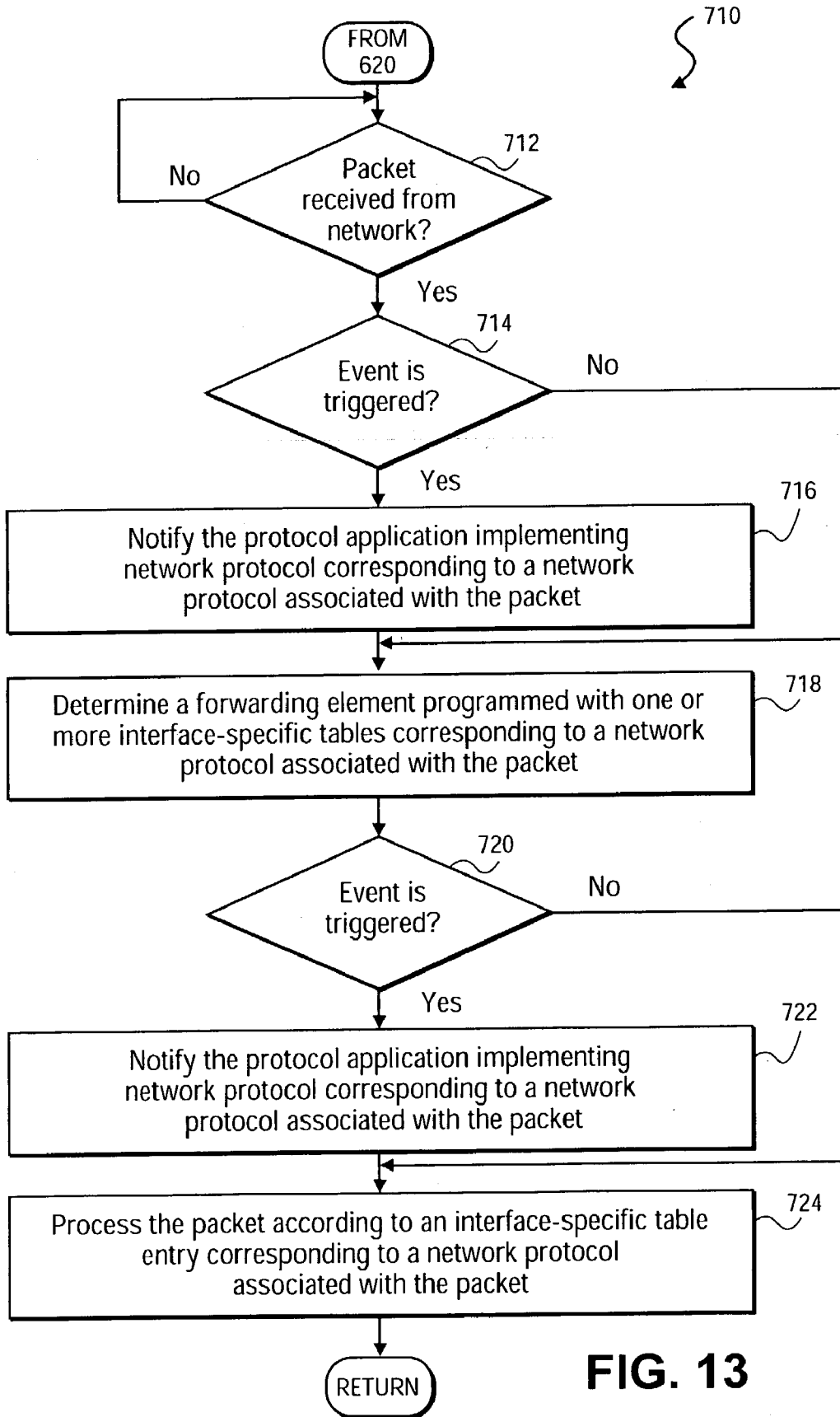


FIG. 13

**APPARATUS AND METHOD FOR CONFIGURING DATA PLANE BEHAVIOR ON NETWORK FORWARDING ELEMENTS**

**FIELD OF THE INVENTION**

[0001] One or more embodiments of the invention relate generally to the field of network communications and protocols. More particularly, one or more of the embodiments of the invention relate to a method and apparatus for configuring data plane behavior on network forwarding elements.

**BACKGROUND OF THE INVENTION**

[0002] Today, numerous independent hardware vendors (IHV) produce networking Applications Specific Integrated Circuits (ASIC) to perform a myriad of packet processing tasks. Unfortunately, the current interface to such ASICs are generally memory mapped registers that have corresponding bit level behavior and documentation. However, not all IHVs limit their products to register level descriptions, some offer C-level or other software interfaces to the hardware. However, these interfaces are merely a convenient reflection of the underlying registers and, therefore, differ from one IHV to another.

[0003] Furthermore, the register level descriptions or their software interfaces to the hardware ASICs make these various products very difficult to use. In fact, these register level models represent a steep learning curve and tight coupling from an original equipment manufacturer (OEM) or an independent software vendor (ISV) that desires to use the ASIC or networking silicon in a product. At such a micro-level description (i.e., the register bits), it is difficult to write code that is reusable across these various ASICs. In addition, it is also difficult to decipher the micro-level functionality of the ASIC's networking silicon.

[0004] A recent trend in the networking industry is the replacement of ASIC, which are relatively inflexible, with more programmable but still performance oriented network devices such as network processors. Unfortunately, network processors are generally in their infancy stages and do not have an abstract programming model or do not have one expressive and flexible enough to grow with advances in the processor itself. Generally, network elements such as switches and routers can be classified into three logical operation components: the control plane, the forwarding plane and the management plane.

[0005] In general, the control plane executes different signaling and/or routing protocols and provides all the routing information to the forwarding plane. The forwarding plane makes decisions based on this information and performs operations on packets such as forwarding classification, filtering and so on. An orthogonal management plane manages the control and forwarding planes. However, the introduction of standardized application programming interfaces (API) within the above-mentioned planes can help system vendors OEMs and end users of these network elements to mix and match components available from different vendors to achieve a device of their choice.

**BRIEF DESCRIPTION OF THE DRAWINGS**

[0006] The various embodiments of the present invention are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which:

[0007] FIG. 1 is a block diagram illustrating a conventional computer network as known in the art.

[0008] FIG. 2 is a block diagram illustrating the conventional computer network as depicted in FIG. 1, further illustrating various network elements utilized to route packets within the network as known in the art.

[0009] FIG. 3 is a block diagram illustrating a conventional network element utilized within the conventional network depicted in FIG. 2.

[0010] FIG. 4 is a block diagram illustrating a network element in accordance with one embodiment of the present invention.

[0011] FIG. 5 is a computer network utilizing network elements as depicted in FIG. 4 in accordance with one embodiment of the present invention.

[0012] FIG. 6 is a flow chart illustrating a method for configuring data plane behavior on network forwarding elements in accordance with one embodiment of the present invention.

[0013] FIG. 7 is a flow chart illustrating a method for processing received configuration information using a control interface corresponding to a network protocol implemented by the received configuration information in accordance with one embodiment of the present invention.

[0014] FIG. 8 is a flow chart illustrating a method for programming one or more data plane forwarding elements according to a network protocol in accordance with one embodiment of the present invention.

[0015] FIG. 9 is a flow chart illustrating a method for programming one or more data plane forwarding elements according to an IPv4 (Internet protocol version 4) routing protocol in accordance with one embodiment of the present invention.

[0016] FIG. 10 is a flow chart illustrating a method for programming one or more data plane forwarding elements according to an IPv6 (Internet protocol version 6) routing protocol in accordance with one embodiment of the present invention.

[0017] FIG. 11 is a flow chart illustrating a method for programming one or more data plane forwarding elements according to an MPLS (Multi-protocol label switching) protocol in accordance with one embodiment of the present invention.

[0018] FIG. 12 is a flow chart illustrating a method for programming one or more data plane forwarding elements according a DiffServ (differentiated services) protocol in accordance with one embodiment of the present invention.

[0019] FIG. 13 is a flow chart illustrating a method for processing a received packet within a forwarding element programmed according to a control plane control interface for implementing a routing protocol in accordance with one embodiment of the present invention.

**DETAILED DESCRIPTION**

[0020] A method and apparatus for configuring data plane behavior on network forwarding elements are described. In one embodiment, the method includes receiving, within a network element control plane, protocol configuration infor-

mation extracted from a network protocol stack (protocol application) utilizing a network protocol application programming interface (API). Once the protocol configuration information is received, the protocol configuration information is processed using a corresponding control interface of the control plane.

[0021] Once the protocol configuration information is processed, the control interface programs one or more data plane forwarding elements of the network element according to the received protocol configuration information. Accordingly, by providing similar control interfaces for multiple network protocols, inter-operability between components from multiple vendors is enabled. As a result, vendor independence is achieved which results in a quicker time to market, improved efficiency, lower costs and greater innovation for future network elements.

[0022] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the embodiments of the present invention. It will be apparent, however, to one skilled in the art that the various embodiments of the present invention may be practiced without some of these specific details. In addition, the following description provides a subset of possible embodiments, and the accompanying drawings thereof, rather than to provide an exhaustive list of all possible implementations of the embodiments of the present invention. In other instances, well-known structures and devices are shown in block diagram form in order to avoid obscuring the details of the various embodiments of the present invention.

#### [0023] System Architecture

[0024] FIG. 1 is a block diagram illustrating a conventional network 100, including an Internet host 102, coupled to host computers 140 (140-1, . . . , 140-N) via the Internet. Generally, the routing of information between Internet host 102 and host computers 140 is performed according to conventional means for transmitting data packets. The various packets are routed within the Internet from Internet host 102 to the various host computers 140. While this model is successful when transmitting conventional packetized information, a variety of applications, including teleconferencing, as well as multimedia applications have emerged, which are sensitive to delays incurred within conventional networks.

[0025] As a result, the emergence of such multimedia and real-time applications, which utilize networks such, as depicted in FIG. 1, have driven the demand for improving, as well as varying quality of service (QoS), switching, routing and the like from available networks. Likewise, various information (packets), which may be routed through network 100, may require various specialized, protocol activity. Unfortunately, networks, as depicted in FIG. 1, utilize a traditional best effort service model which does not pre-allocate bandwidth for handling network traffic, but simply routes packets utilizing current available bandwidth on a best effort basis. As a result, varying queuing delays and congestion across the Internet are quite common.

[0026] The best effort service model implemented by network 100 is based on the transmission control protocol (TCP) Internet protocol (IP) (TCP/IP). Specifically, the protocol implementation refers to Internet protocol version

4 (IPv4). In general, IPv4 is the most widely installed level of Internet protocol in use today. As known to those skilled in the art, an IP address is a 32-bit number that identifies each sender or receiver of information that is sent in packets across the Internet. An IP address generally has two parts: an identifier of a particular network on the Internet; and an identifier of the particular device within the network.

[0027] As such, the Internet is an interconnection of many individual networks, and IPv4 is a set of rules for one network communicating with another, provided that the network knows its own Internet address and that of any other networks with which it communicates. Unfortunately, the 32-bit IP address specified by IPv4 is very close to being exceeded. In other words, the Internet's growth makes it likely that without some new architecture, the number of possible network addresses using the addressing scheme provided by IPv4 may soon be exceeded. Likewise, the best effort service model provided by current TCP/IP is insufficient for various current, as well as future networking applications.

[0028] Accordingly, a next generation version of the Internet protocol has been implemented. As known to those skilled in the art, this next generation Internet protocol is referred to as "Internet protocol version 6" (IPv6). The most obvious improvement in IPv6 over IPv4 is that the IP addresses are lengthened from 32-bits to 128-bits. As a result, this extension anticipates considerable future growth of the Internet and provides relief for what was perceived as an impending shortage of network addresses. Furthermore, IPv6 describes rules for three types of addressing: uni-cast (one host to another host); any cast (one host to the nearest of multiple hosts); and multicast (one host to multiple hosts).

[0029] In addition, IPv6 provides various enhancements. For example, IPv6 enables an identification mechanism for packets deemed to require a higher QoS relative to other network traffic packets. In other words, packets belonging to an end-to-end path (or "flow") can be identified as a part of, for example, a multimedia presentation that requires real-time arrival. As a result, by providing this identification, packets requiring real-time delivery or which are sensitive to delay may be given a higher QoS relative to standard network traffic.

[0030] For example, FIG. 2 is a block diagram further illustrating the network of FIG. 1 depicted to illustrate the various network elements between, for example, a real-time application server (source computer) 200, and a destination computer 220. Due to the fact that the network is configured according to, for example, the IPv4 routing protocol, received network traffic is forwarded based on the best effort service model. As a result, transmission of packetized data between the real-time application server 200 and a destination computer 220 incurs various routing delays due to the fact that congestion across the Internet is quite common as a result of the increasing widespread use of the Internet. For example, during peak hours, transmission of real-time packetized data would incur such substantial delays as to render the application unusable at the destination source computer.

[0031] A current network protocol (model) for providing service differentiation (varying QoS) for traffic flows is the differentiated services (DiffServ) model. The DiffServ model provides qualitative differentiation to flow aggregates unlike the hard guarantees such as delay bounds and assured

bandwidth provided by the integrated service (IntServ) model. For example, the qualitative differentiation provided by the DiffServ model may be configured to provide Class A higher priority than Class B. Furthermore, the DiffServ model provides data path elements (DPE) including, but not limited to, classifiers, meters, droppers, queues and schedulers.

[0032] A further network protocol which may be utilized to, for example, speed up network traffic in order to implement applications which are very sensitive to queuing delays and congestion within the network such as, for example, audio and voice applications, is the multi-protocol label switching protocol (MPLS). The MPLS protocol is a standard approved technology for speeding network traffic flow, as well as simplified management thereof. MPLS involves setting up a specific path for a given sequence of packets identified by a label put into each packet. As a result, time is saved which is generally required for a router to look up the address to the next node to forward the packet.

[0033] In addition, MPLS is referred to as a multi-protocol because it works with the Internet protocol, asynchronous transfer mode (ATM) protocol and the frame relay network protocol, as known in the art. Furthermore, MPLS allows most packets to be forwarded at the layer-2 (switching) level, rather than the layer-3 (routing) level. Moreover, in addition to moving traffic faster, MPLS provides simplified management of a network for quality of service (QoS) and Traffic Engineering. Accordingly, networks implementing the MPLS protocol are able to adequately provide sufficient quality of service to different mixtures of network traffic. As described herein, the various routing, switching and QoS protocols described above are collectively referred to as "network protocols."

[0034] Referring now to FIG. 3, FIG. 3 is a conventional network element 302 as utilized within network 300, as shown in FIG. 2. As illustrated, the network element includes a forwarding plane 380 which uses an ingress filter block 304, a forwarding decision block 390 and an egress filter block 306 to route incoming packets to a destination, as dictated by the control plane 310. Unfortunately, network element 302 provides tightly coupled forwarding and control planes within a single device. As a result, implementing varying QoS, as required by delay sensitive applications, within network element 302 generally requires intimate knowledge of a proprietary interface used within the device.

[0035] However, a recent trend in the networking industry is the replacement of conventional network elements, which are relatively inflexible, with more programmable, but still performance oriented, network devices such as network processors. Unfortunately, network processors are generally in their infancy stages and do not have an abstract programming model or do not have one expressive and flexible enough to grow with advances in the processor itself.

[0036] Generally, network elements such as switches and routers can be classified into three logical operation components: the control plane, the forwarding plane and the management plane. For example, the control plane executes different signaling or routing protocols and provides all routing information to the forwarding plane. The forwarding plane makes decisions based on this information and performs operations on packets such as forwarding, classification, filtering and so on. An orthogonal management plane manages the control and forwarding planes.

[0037] However, the introduction of standardized application programming interfaces (API) within the above-mentioned planes can help system vendors, OEMs and end users of these network elements to mix and match components available from different vendors to achieve a device of their choice. Accordingly, FIG. 4 is a block diagram illustrating a network element 400 comprised of a separate control plane 410 and data plane 430 coupled together via interconnect 402, in accordance with one embodiment of the present invention.

[0038] In one embodiment, the control interface 410 enables queries of the data plane for determining information required to program the forwarding element of the data plane 430. This allows for a larger degree of generality in controlling the specific behavior in network forwarding elements. As described herein, the various parameters and information for generating a tables and populating their entries for implementing the respective protocol, is collectively referred to herein as "protocol configuration information."

[0039] Accordingly, control interface 420 is able to program a selected data plane forwarding element to implement third party network protocol stacks (protocol applications) within data plane 430, regardless of the vendor source of the data plane. In one embodiment, the control interface 420 programs the data plane 430 according to protocol configuration information extracted from the protocol application by an API corresponding with the protocol. For example, network element 400 is illustrated as implementing the IPv4, IPv6 routing protocols, as well as the MPLS and DiffServ network protocols ("network protocols"), via applications 424 and 426. Likewise, management/configuration application 422 is also provided.

[0040] In order to enable the various protocol applications from different third party developers, one embodiment of the present invention provides APIs for, for example, the IPv4 routing protocol, IPv6 routing protocol, MPLS network protocol and DiffServ network protocol. However, various APIs for additional protocols desired may be implemented in accordance with embodiments of the present invention. Accordingly, by utilizing an API in accordance with embodiments of the present invention, a third party protocol applications may be loaded with a vendor A control plane to implement a specific protocol as desired by the third party software developer within a vendor B data plane.

[0041] In one embodiment, communication interfacing, as well as implementation of the protocol within the data plane 430, is handled by the respective protocol API. In other words, control interface 420 exposes APIs for the protocols described above. As a result, incorporating a corresponding API into a conventional protocol application provides platform independent protocol applications. As such, in one embodiment, the corresponding API provides protocol configuration information extracted from the protocol application to control plane 420.

[0042] In one embodiment, the control plane 420 uses this protocol configuration information to program the data plane 430 forwarding elements to implement the protocol. In addition, communication between the control interface 420 and the data plane 430 is achieved using, for example, an interconnect message protocol, or the like. For example, the IPv6 routing protocol may be implemented within forward-

ing element 440. As illustrated, IPv6 forwarding tables, as well neighbor table 460 is formed within forwarding element 440 during device boot-up. In addition, as dictated by the protocol configuration information, the interface-specific table entries (462 and 464) are generated and populated within the respective table (460) in order to dictate routing of received packets via interfaces 442.

[0043] Likewise, MPLS, FTN (forward equivalence class (FEC) to next hop label-forwarding entry (NHLFE)) and incoming label map (ILM) tables may also be implemented within forwarding element 440 with specific entries (452, 454 and 456). The specific entries are dictated by the corresponding Label Distribution Protocol (LDP) application in order to control behavior of forwarding elements when receiving corresponding packets via interfaces 442. Likewise, the IPv4 routing protocol, as well as the DiffServ routing protocol may be implemented within forwarding element 470.

[0044] This capability is provided due to the fact that in one embodiment, the control interface 420 provides the mechanism to target one of many specific elements within the data plane 430. That is, a single, specific table entry within the one or more tables within one of multiple forwarding elements may be addressed. Likewise, in one embodiment, the control interface provides a mechanism for control plane components (protocol applications) to query the forwarding elements to determine parameters and capabilities of the data plane.

[0045] Accordingly, FIG. 5 is a block diagram illustrating a network 500 utilizing network elements 400, as depicted in FIG. 4, in accordance with one embodiment of the present invention. As illustrated, a real-time application server 200 is able to receive improved/varying QoS provided by, for example, the DiffServ network protocol within network element 400. Likewise, other various source computers may require a separate routing/network protocol such as, for example, IPv4 (for legacy applications) and IPv6 routing protocols, MPLS network, or the like. These protocols may be implemented within the network element using, for example, third party protocol applications including a corresponding protocol API according to an embodiment of the present invention.

[0046] Furthermore, the control interface 420 within network elements 400 is designed to be completely asynchronous in nature to support any kind of interconnect technology for interconnect 402 between the control and data planes. Control planes and data planes can be connected using network technologies such as Ethernet, cable line, a bus or optical fiber. Likewise, the connection between the control and data plane could be accomplished via bus technology such as compact PCI (peripheral component interconnect) or the control and data plane could be co-located and connected via interprocess communications. Procedural methods for implementing embodiments of the present invention are now described.

[0047] Operation

[0048] FIG. 6 is a flow chart illustrating a method 600 for configuring data plane behavior with the network forwarding elements of, for example, network element 400, as depicted in FIG. 4 in accordance with one embodiment of the present invention. At process block 602 protocol con-

figuration information, extracted from a protocol application, is received within a network element control plane. In one embodiment, the protocol application implements a network/routing protocol utilizing a protocol application programming interface (API) which is responsible for extracting the protocol configuration information.

[0049] In one embodiment, the protocol API supports network protocols such as, for example, the DiffServ network protocol, MPLS network protocol, as well as IPv4 and IPv6 routing protocols. Accordingly, implementation of the respective protocol specified by the received protocol configuration information is hidden from the application programmer. Once the protocol configuration information application is received, at process block 604 the received protocol configuration information is processed using a control interface corresponding to the protocol implemented by the received protocol application.

[0050] For example, as depicted with reference to FIG. 4, control interface 420 is comprised of control interfaces IPv4, IPv6, MPLS and DiffServ control interfaces. As a result, control interface 420 determines a corresponding protocol interface for processing the application received from, for example, a third party developer. As such, once selected, at process block 620, one or more data plane forwarding elements of the network element are programmed according to the protocol configuration information extracted from the protocol application, for example, as illustrated in FIG. 4.

[0051] FIG. 7 is a flow chart illustrating a method 606 for processing of received protocol configuration information at process block 604, as depicted in FIG. 6, in accordance with one embodiment of the present invention. At process block 608, the control interface selects a forwarding element within the data plane to implement the network routing or non-routing protocol associated with the received protocol configuration information. For example, control interface 420 could select forwarding element 440 to implement the IPv6 routing protocol.

[0052] Next, at process block 610, one or more interface-specific tables are determined that require formation within the selected forwarding element. Once determined at process block 612, one or more entries within the interface-specific (or protocol-specific) tables are determined by the control interface, according to the protocol configuration information. As a result, a forwarding element containing the interface-specific tables determines routing decisions according to entries within the interface-specific tables. For example, entries 1-3 of MPLS table 450 in addition to entries 1 and 2 of IPv6 table 460 are determined.

[0053] FIG. 8 is a flow chart illustrating a method 622 for programming the data plane forwarding elements of process block 620 as depicted in FIG. 6, in accordance with one embodiment of the present invention. At process block 624, after querying the protocol capability provided by the data plane, the network protocol specified by the protocol configuration information is registered with the data plane during device boot-up to enable receipt of events from the data plane. In other words, during various processing within the forwarding element, the implemented protocol may trigger an event that requires communication to the control plane in order to determine subsequent processing behavior within the data plane.

[0054] At process block 626 the control interface selects the forwarding element from one or more forwarding ele-

ments of the network data plane such as, for example, forwarding element 440 (FIG. 4). Next, at process block 628 one or more interface-specific tables, generated within the selected forwarding element at device boot-up, are selected. Once selected at process block 630, the control interface generates one or more entries within the interface-specific tables of the selected forwarding element, as dictated by the protocol configuration information extracted from a received protocol application by a corresponding protocol API.

[0055] FIG. 9 is a flow chart for a method 632 of programming data plane behavior of process block 620, as depicted in FIG. 6, in order to implement an IPv4 routing protocol in accordance with one embodiment of the present invention. At process block 634 the IPv4 routing protocol is registered with the network element data plane, during device boot-up, to enable receipt of events from the data plane. At process block 636, an IPv4 forwarding table is generated within the selected forwarding element within the selected forwarding element during device boot-up. At process block 638 an address resolution (ARP) table is generated within the selected forwarding element during device boot-up.

[0056] Next, at process block 640, a forwarding element within the data plane is selected to implement an IPv4 routing protocol. At process block 642 it is determined whether a request is received from the protocol application, which may include route update requests and ARP update request. When a route request is received, at process block 644, IPv4 routing entries are either added or deleted within the forwarding table according to the received update request. When an ARP request is received, at process block 646, IPv4 entries are either added or deleted within the address resolution table according to the received request. Accordingly, process blocks 642-646 are repeated until programming of the forwarding elements is complete.

[0057] FIG. 10 depicts a flow chart illustrating a method 650 for programming the data plane of process block 620, as depicted in FIG. 6, in order to implement an IPv6 routing protocol in accordance with one embodiment of the present invention. At process block 652, the IPv6 routing protocol is registered within the data plane, during device boot-up, to enable receipt of events from the data plane, as indicated above. At process block 654 an IPv6 forwarding table generated within the selected forwarding element during device boot-up. Once selected at process block 656, an IPv6 neighbor table is generated within the selected forwarding element during device boot-up.

[0058] Next, at process block 658, a forwarding element is selected from the network element data plane. At process block 660 it is determined whether a request is received from the protocol application, which may include route update requests and neighbor table update requests. When a route request is received, at process block 662 IPv6 routing entries are either added or deleted within the forwarding table according to the received request. When a neighbor table update request is received, at process block 664 IPv6 neighbor entries are either added or deleted in the neighbor table according to the received request. Accordingly, process blocks 660-664 are repeated until programming of the forwarding elements is complete.

[0059] FIG. 11 is a flow chart illustrating a method 670 for programming forwarding elements of the data plane at

process block 620, as depicted in FIG. 6, to implement an MPLS network protocol in accordance with one embodiment of the present invention. At process block 672, an FTN table is generated within a forwarding element selected, during device boot-up, for implementing the MPLS switching protocol. At process block 674, an ILM table is generated within the selected forwarding element during device boot-up. Next, at process block 676, an FTN DiffServ context table is generated within the selected forwarding element during device boot-up.

[0060] Once generated, at process block 678, an ILM DiffServ context table is generated within the selected forwarding element during device boot-up. Once generated, at process block 680, it is determined whether a protocol application request is received. When an FTN request is received, process blocks 682 and 684 are performed. However, when an ILM request is received, process blocks 686 and 688 are formed. Otherwise, control flow returns to process block 620 (FIG. 6), which generally occurs once programming of the data plane forwarding elements is complete by repeating of process blocks 680-688 for each protocol application request.

[0061] Accordingly, at process block 682 MPLS FTN entries are either added or deleted within the FTN table when an MPLS FTN entry request is received. Likewise, at process block 684, MPLS E-LSP (explicit route label switch path) configuration entries are added or deleted in the FTN DiffServ context tables when an MPLS E-LSP configuration entry request is received. Otherwise, at process block 686 MPLS ILM entries are either added or deleted within the ILM table when an MPLS ILM entry request is received. Likewise, at process block 688, MPLS E-LSP pre-configuration entries are added or deleted within the ILM DiffServ context table when an MPLS E-LSP pre-configuration entry request is received.

[0062] FIG. 12 is a flow chart illustrating a method 690 for programming data plane forwarding elements of process block 620, as depicted in FIG. 6, to implement a DiffServ network protocol in accordance with one embodiment of the present invention. At process block 692, a plurality of DiffServ tables are generated within the forwarding element selected to implement the DiffServ routing protocol, for example, DiffServ DPE, during device boot-up. In one embodiment, each call (protocol application request) may include entries of one of the following types: filter, meter, marker, queue or scheduler. At process block 694, filter entries are either added or deleted in a DiffServ classifier table when a classifier table entry request is received. At process block 696, meter entries within a DiffServ meter table are either added, modified or deleted when a meter entry request is received.

[0063] Furthermore, at process block 698, marker entries within a DiffServ marker table are added, modified or deleted when a marker table entry request is received. At process block 700, dropper entries are added, modified or deleted within a DiffServ algorithmic dropper table generated within the selected forwarding element when a dropper entry request is received. At process block 702 query entries within a DiffServ queue are added, modified or deleted when a queue entry request is received. Finally, at process block 704, scheduler entries within a DiffServ scheduler table are either added, modified or deleted when a scheduler table entry request is received.

[0064] Finally, FIG. 13 is a flow chart illustrating a method 710 for behavior of, for example, a network element implementing a control interface in accordance with one embodiment of the present invention. At process block 712 it is determined whether a packet is received. Once received at process block 714, it is determined whether an event is triggered by the received packet, such as a "protocol unsupported" event. When an event is triggered process block 716 is performed. At process block 716 a forward element application, implementing the network protocol corresponding to the received packet, is notified of the event. In one embodiment, notification may include forwarding of the received packet to a responsible forwarding element.

[0065] Otherwise at process block 718, a forwarding element, programmed with one or more interface-specific tables corresponding to the network protocol associated with the packet, is determined. Once determined, at process block 720, it is determined whether an event is triggered by table look-up, such as a "missing entry/no entry found" event. When an event is triggered process block 722 is performed. At process block 722 a forward element application, implementing the network protocol corresponding to the received packet, is notified of the event. In one embodiment a "no entry found" event could cause a DROP action of the received packet by the forwarding element.

[0066] Otherwise, at process block 724 the forwarding element processes the packet according to an interface-specific table entry corresponding to the network protocol associated with the packet and populated according to corresponding protocol configuration information. In one embodiment, communication between the control plane and the data plane is provided via, for example, an interconnect messaging protocol as known in the art. In alternative embodiments, additional messaging or communication protocols between the control plane and data plane may be utilized.

[0067] Accordingly, utilizing embodiments of the present invention third party protocol application development for, for example, network processors may be implemented regardless of the vendor source of either the control plane or data plane elements of the forwarding elements such as a network processor. In other words, use of control interfaces and corresponding APIs, in accordance with embodiments of the present invention, allow interoperability between components of multiple vendors for implementing network elements to provide improved, as well as varying quality of service to a plurality of varying network traffic.

[0068] Accordingly, by isolating network/routing protocol implementations within data planes from third party protocol application developers, platform independence is achieved which results in quicker time to market of products, improved efficiency and lower costs and greater innovation. Furthermore, the control interface provided is designed to be completely asynchronous in nature to support any kind of internet interconnect technology between the control and data plane. As a result, control and data planes can be connected using network technologies such as, for example, Ethernet fiber.

[0069] Likewise, the connection (interconnect link) between the control and data planes can be made via bus technology such as compact PCI or the control and data plane can be co-located and connected via interprocess

communication. Furthermore, the control interfaces provide an easy to use, scaleable and robust interface to configure IPv4/IPv6/MPLS/DiffServ behavior in a network processor data plane. In one embodiment a network processor, for example, as manufactured by the Intel Corporation of Santa Clara, Calif., is used for the management, applications and routing/MPLS/QoS applications running in the control plane. Furthermore, integration of third party routing stacks and management applications are also easily implemented utilizing a control interface in accordance with embodiments of the present invention.

#### [0070] Alternate Embodiments

[0071] Several aspects of one implementation of the control interface and API for configuring multiple data plane behavior have been described. However, various implementations of the control interface and API provide numerous features including, complementing, supplementing, and/or replacing the features described above. Features can be implemented as part of the control plane or as part of a network element in different embodiment implementations. In addition, the foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the embodiments of the invention. However, it will be apparent to one skilled in the art that the specific details are not required in order to practice the embodiments of the invention.

[0072] In addition, although an embodiment described herein is directed to a control interface and APIs for a network element, it will be appreciated by those skilled in the art that the embodiments of the present invention can be applied to other systems. In fact, systems for supporting multiple protocols with a network element fall within the embodiments of the present invention, as defined by the appended claims. The embodiments described above were chosen and described in order to best explain the principles of the embodiments of the invention and its practical applications. These embodiments were chosen to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated.

[0073] It is to be understood that even though numerous characteristics and advantages of various embodiments of the present invention have been set forth in the foregoing description, together with details of the structure and function of various embodiments of the invention, this disclosure is illustrative only. In some cases, certain subassemblies are only described in detail with one such embodiment. Nevertheless, it is recognized and intended that such subassemblies may be used in other embodiments of the invention. Changes may be made in detail, especially matters of structure and management of parts within the principles of the embodiments of the present invention to the full extent indicated by the broad general meaning of the terms in which the appended claims are expressed.

[0074] Having disclosed exemplary embodiments and the best mode, modifications and variations may be made to the disclosed embodiments while remaining within the scope of the embodiments of the invention as defined by the following claims.

What is claimed is:

1. A method comprising:
  - receiving, within a network element control plane, protocol configuration information extracted from a protocol application utilizing a network protocol application programming interface (API);
  - processing the received protocol configuration information using a control interface corresponding to a network protocol specified by the received protocol configuration information; and
  - configuring, by the control interface, one or more data plane forwarding elements of the network element according to the received protocol configuration information.
2. The method of claim 1, wherein processing the received protocol configuration information further comprises:
  - selecting a forwarding element within the data plane to implement the network protocol specified by the protocol configuration information;
  - determining one or more interface-specific tables to form within the selected forwarding element according to the protocol configuration information; and
  - determining one or more entries within the interface-specific tables such that a forwarding element containing the interface-specific tables determines decisions according to entries within the interface-specific tables.
3. The method of claim 1, wherein configuring the one or more data plane forwarding elements further comprises:
  - registering the network protocol specified by the protocol configuration information with the data plane to enable receipt of events from the data plane;
  - selecting a forwarding element from one or more forwarding elements of a network element data plane;
  - querying the selected forwarding element for one or more interface-specific tables within the selected forwarding element according to the protocol configuration information; and
  - generating and populating one or more entries within the interface-specific tables of the selected forwarding element in order to control data plane behavior when processing packets corresponding to the network protocol implemented by the protocol application.
4. The method of claim 3, wherein generating further comprises:
  - querying the selected forwarding element to determine the generated interface-specific tables; and
  - populating the generated interface-specific tables according to the received protocol configuration information.
5. The method of claim 1, wherein configuring the data plane elements further comprises:
  - registering the network protocol with the network element data plane to enable receipt of events from the data plane during boot-up;
  - selecting a forwarding element within the data plane of the network element to implement an IPv4 routing protocol according to the received protocol configuration information;
  - querying the selected forwarding element for a forwarding table generated within the selected forwarding element during boot-up;
  - querying the selected forwarding element for an address resolution table generated within the selected forwarding element during boot-up;
  - adding/deleting IPv4 routing entries within the forwarding table when a protocol application route update request is received; and
  - adding/deleting IPv4 entries in the address resolution table when a protocol application address resolution table update request is received.
6. The method of claim 1, wherein configuring the one or more data plane forwarding elements further comprises:
  - registering the network protocol with the data plane to enable receipt of events from the data plane during boot-up;
  - selecting a forwarding element of the network element data plane to implement an IPv6 routing protocol according to the received protocol configuration information;
  - querying the selected forwarding element for an IPv6 forwarding table generated within the selected forwarding element during boot-up;
  - querying the selected forwarding element for an IPv6 neighbor table generated within the selected forwarding element during boot-up;
  - adding/deleting IPv6 routing entries in the forwarding table when a protocol application route update request is received; and
  - adding/deleting IPv6 neighbor entries in the neighbor table when a protocol application neighbor table update request is received.
7. The method of claim 1, wherein configuring the one or more forwarding elements of the data plane further comprises:
  - generating an FTN table within a forwarding element selected for implementing an MPLS protocol during boot-up;
  - generating an ILM table within the selected forwarding element during boot-up;
  - generating an FTN DiffServ context table within the selected forwarding element during boot-up;
  - generating an ILM DiffServ context table within the selected forwarding element during boot-up;
  - adding/deleting MPLS FTN entries within the FTN table when a protocol application FTN table update request is received;
  - adding/deleting MPLS ILM entries within the ILM table when a protocol application ILM table update request is received;
  - adding/deleting MPLS E-LSP configuration entries in the FTN DiffServ context table when a protocol application MPLS E-LSP configuration update request is received; and

adding/deleting MPLS E-LSP pre-configuration entries in the ILM DiffServ context table when a protocol application MPLS E-LSP pre-configuration update request is received.

8. The method of claim 1, wherein configuring the one or more forwarding elements of the data plane further comprises:

generating a plurality of DiffServ tables within a forwarding element selected to implement a DiffServ network protocol during boot-up;

adding/deleting filter entries in a DiffServ classifier table generated within the selected forwarding element when a protocol application classifier update request is received;

adding/modifying/deleting meter entries within a DiffServ meter table generated within the selected forwarding element when a protocol application meter update request is received;

adding/modifying/deleting marker entries within a DiffServ marker table generated within the selected forwarding element when a protocol application marker update request is received;

adding/modifying/deleting dropper entries within a DiffServ algorithmic dropper table generated within the selected forwarding element when a protocol application dropper update request is received;

adding/modifying/deleting query entries within a DiffServ queue table generated within the selected forwarding element when a protocol application queue update request is received; and

adding/modifying/deleting scheduler entries within a DiffServ scheduler table generated within a selected forwarding element when a protocol application scheduler update request is received.

9. The method of claim 1, further comprising:

receiving, within the data plane, a packet;

when an event is triggered, notifying a protocol application implementing a network protocol corresponding to the received packet;

otherwise, determining a forwarding element programmed with one or more interface-specific tables corresponding to a network protocol associated with the received packet;

when an event is triggered, notifying a protocol application implementing a network protocol corresponding to the received packet; and

otherwise, processing the received packet according to an interface-specific table entry generated by the control interface according to the protocol configuration information and associated with the received packet.

10. The method of claim 1, wherein the network protocol API is one of a routing protocol API, a switching protocol API, and a quality-of-service (QoS) protocol API.

11. The method of claim 10, wherein the routing protocol API is one of an IPv4 protocol and an IPv6 protocol.

12. The method of claim 10, wherein the switching protocol API is an MPLS protocol API.

13. The method of claim 10, wherein the QoS protocol API is a DiffServ protocol API.

14. The method of claim 1, wherein the network protocol API comprises quality-of-service capabilities to enable associating packets of a particular traffic flow with a desired special handling.

15. The method of claim 14, wherein the desired special handling comprises at least one of real-time service or non-default quality-of-service.

16. A computer readable storage medium including program instructions that direct a computer to perform a method when executed by a processor, the method comprising:

receiving, within a network element control plane, protocol configuration information extracted from a protocol application utilizing a network protocol application programming interface (API);

processing the received protocol configuration information using a control interface corresponding to a network protocol specified by the received protocol configuration information; and

configuring, by the control interface, one or more data plane forwarding elements of the network element according to the received protocol configuration information.

17. The computer storage medium of claim 16, wherein processing the received software further comprises:

selecting a forwarding element within the data plane to implement the network protocol specified by the protocol configuration information;

determining one or more interface-specific tables to form within the selected forwarding element according to the protocol configuration information; and

determining one or more entries within the interface-specific tables such that a forwarding element containing the interface-specific tables determines decisions according to entries within the interface-specific tables.

18. The computer readable storage medium of claim 16, wherein configuring the one or more data plane forwarding elements further comprises:

registering the network protocol specified by the protocol configuration information with the data plane to enable receipt of events from the data plane during boot-up;

selecting a forwarding element from one or more forwarding elements of a network element data plane;

querying the selected forwarding element for one or more interface-specific tables within the selected forwarding element according to the protocol configuration information; and

generating and populating one or more entries within the interface-specific tables of the selected forwarding element in order to control data plane behavior when processing packets corresponding to the network protocol implemented by the protocol application.

19. The computer readable storage medium of claim 16, wherein the method further comprises:

receiving, within the data plane, a packet;

when an event is triggered, notifying a protocol application implementing a network protocol corresponding to the received packet;

otherwise, determining a forwarding element programmed with one or more interface-specific tables corresponding to a network protocol associated with the received packet;

when an event is triggered, notifying a protocol application implementing a network protocol corresponding to the received packet; and

otherwise, processing the received packet according to an interface-specific table entry generated by the control interface according to the protocol configuration information and associated with the received packet.

**20.** The method of claim 16, wherein the network protocol API is one of a routing protocol API, a switching protocol API, and a quality-of-service (QoS) protocol API.

**21.** The method of claim 20, wherein the routing protocol API is one of an IPv4 protocol and an IPv6 protocol.

**22.** The method of claim 20, wherein the switching protocol API is an MPLS protocol API.

**23.** The method of claim 20, wherein the QoS protocol API is a DiffServ protocol API.

**24.** The method of claim 16, wherein the network protocol API comprises quality-of-service capabilities to enable associating packets of a particular traffic flow with a desired special handling.

**25.** The method of claim 24, wherein the desired special handling comprises at least one of real-time service or non-default quality of service.

**26.** A network element, comprising:

a data plane including one or more forwarding elements to process received packets according to a corresponding interface-specific table contained within a respective forwarding element;

a control plane coupled to the data plane via an interconnect link, the control plane including one or more protocol applications implementing network protocols utilizing a network protocol application programming interface (API); and

a control interface used to generate and configure the interface-specific tables contained within the data plane according to protocol configuration information extracted from the protocol applications of the control plane by a respective protocol API.

**27.** The network element of claim 26, wherein the control interface is further caused to select a forwarding element within the data plane to implement the network protocol specified by protocol configuration information; determine one or more interface-specific tables to be formed within the selected forwarding element according to extracted protocol configuration information; and determine according to protocol configuration information, one or more entries within the interface-specific tables such that a forwarding element containing the interface-specific tables determines protocol decisions according to entries within the interface-specific tables.

**28.** The network element of claim 26, wherein the control interface is further caused to register the network protocol specified by the protocol configuration information with the data plane to enable receipt of events from the data plane

during boot-up; select a forwarding element from one or more forwarding elements of a network element data plane; query the selected forwarding element for one or more interface-specific tables generated within the selected forwarding element during boot-up; and generate and populate one or more entries within the interface-specific tables of the selected forwarding element according to extracted protocol configuration information.

**29.** The network element of claim 26, wherein the control interface is further caused to receive, within the data plane, a packet; determine a forwarding element programmed with one or more interface-specific tables corresponding to a network protocol associated with the packet; process the packet according to an interface-specific table entry generated by the control interface according to received protocol configuration information; and when an event is triggered, notify the protocol application implementing a network protocol associated with the packet.

**30.** The network element of claim 26, wherein the interconnect link is one of a bus, a cable, an optical fiber, and an Ethernet link.

**31.** A system comprises:

a network including a plurality of coupled network elements, comprising:

a data plane including one or more forwarding elements to process received packets according to a corresponding interface-specific table contained within a respective forwarding element;

a control plane coupled to the data plane via an interconnect link, the control plane including one or more protocol applications implementing network protocols utilizing a network protocol application programming interface (API); and

a control interface used to generate and configure the interface-specific tables contained within the data plane according to protocol configuration information extracted from the protocol applications of the control plane by a respective protocol API.

**32.** The system of claim 31, wherein the control interface is further caused to select a forwarding element within the data plane to implement the network protocol specified by protocol configuration information; determine one or more interface-specific tables to be formed within the selected forwarding element according to extracted protocol configuration information; and determine according to protocol configuration information, one or more entries within the interface-specific tables such that a forwarding element containing the interface-specific tables determines protocol decisions according to entries within the interface-specific tables.

**33.** The system of claim 31, wherein the control interface is further caused to register the network protocol specified by the protocol configuration information with the data plane to enable receipt of events from the data plane during boot-up; select a forwarding element from one or more forwarding elements of a network element data plane; query the selected forwarding element for one or more interface-specific tables generated within the selected forwarding element during boot-up; and generate and populate one or more entries within the interface-specific tables of the selected forwarding element according to extracted protocol configuration information.

34. The system of claim 31, wherein the control interface is further caused to receive, within the data plane, a packet; determine a forwarding element programmed with one or more interface-specific tables corresponding to a network protocol associated with the packet; process the packet according to an interface-specific table entry generated by the control interface according to received protocol configu-

ration information; and when an event is triggered, notify the protocol application implementing a network protocol associated with the packet.

35. The system of claim 31, wherein the interconnect link is one of a bus, a cable, an optical fiber and an Ethernet link.

\* \* \* \* \*