Title: ROUTING TABLE MANAGER USING PSEUDO ROUTES

Abstract: Routing data packets using pseudo routes. Preferred methods include the steps (i) generating a routing table having a plurality of route records, wherein each route record has a routing prefix entry and a corresponding next hop entry; for each route in the routing table having a recursively-resolvable next hop, generating a pseudo-route record in the routing table for a pseudo route, wherein the pseudo-route record comprises the corresponding next hop entry and recursively-resolved link-layer routing data for that next hop entry; and, (iii) routing packets using the routing table. Thus, the routing table will contain the routes available to the router (preferably provided by various routing protocols), and will also contain a pseudo route for each next hop address that is not directly resolvable. In this manner, route updates affecting recursively-resolved routes may be accomplished by simply changing the link-layer routing data of the pseudo route associated with the next hop of the recursively-resolved route.
For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
Description

ROUTING TABLE MANAGER USING PSEUDO ROUTES

FIELD OF INVENTION

[1] The present invention relates to processing data packets at a packet switch or router in a packet switched communications network, and more particularly, to a method of routing packets using routes that are recursively resolved.

BACKGROUND

[2] A router within a data network receives data packets from the network via one or more physical ports, and processes each data packet primarily to determine on which outgoing port the packet should be forwarded. The router typically includes a Routing Table Manager (RTM) to combine routing information (routes) from multiple routing protocols to generate an integrated master routing table. A route is composed of a destination prefix and next-hop information. The destination prefix specifies a destination address range in the network protocol (e.g., the internet protocol, or IP) address space and is identified by the address and the number of unmasked bits counting from the left. The next-hop is the destination address or the interface where the packet should be forwarded to reach its final destination.

When a next-hop is reachable within a single hop across a link layer, it is said to be adjacent. If a route resolves over an adjacent next-hop it is said to be directly resolvable. If the next-hop itself resolves over another route, the route is said to be recursively resolved. Inter Domain Routing Protocols such as BGP (Border Gateway Protocol) and EGP (Exterior Gateway Protocol) typically recursively resolve its routes over other routes installed by Internal Gateway Protocols (IGP) such as O SPF (Open Shortest Path First), RIP (Routing Information Protocol) and IS-IS (Intermediate System - Intermediate System).

[4] The router typically generates a forwarding table (FT) from the master routing table. Each entry in the forwarding table points to either other routes in the forwarding table or to a forwarding entry specifying the egress interface and other information such as the link layer address. For each packet received, the Forwarding Engine (FE) selects a matching route from the forwarding table according to the longest prefix match algorithm, i.e. the route indexed by the prefix with the largest number of leading address bits matching those of the packet destination address. The FE uses the information specified in the forwarding entry corresponding to the matching route to forward the packet towards its final destination.

[5] Prior art implementations have handled recursively-resolved routes in a number of ways. The simplest approach is to recursively resolve the next-hop of each route in the table in order to generate a forwarding entry and then downloading the resulting link-
layer forwarding data into the forwarding table. In this implementation, every route has a corresponding FT entry having link-layer forwarding data. The disadvantage of this approach is that if a route update causes one or more next-hops to be resolved differently, all routes resolving over those next-hops have to be repopulated in the forwarding table.

Another approach consists in downloading into the forwarding table an exact copy of the master routing table including its recursive dependencies. The forwarding entry of a recursively resolved route points back to the route resolving the next-hop. Therefore when a route is updated only the forwarding entry for the changed route must be repopulated in the forwarding table. The disadvantage of this approach is the additional complexity of the FE, which must be able to perform multiple look-ups on a packet in order to recursively resolve the destination address before being able to forward it to its destination. If the forwarding table is implemented in hardware this leads to an increase in cost and/or a degradation in performance.

Another approach used in the prior art consists of decoupling the forwarding entry information from the routes and storing it in a separate table whose entries are pointed to by the routes in the FT. That is, each route in the forwarding table provides a pointer to a forwarding entry in the separate forwarding table. This is depicted in Fig. IA, where the routing table 102 is transformed into forwarding table 104 containing prefixes and pointers to the separate forwarding table 106, which contains the link-layer forwarding entries. (Note that the pointers of table 104 are indicated by the lines connecting the route entry of table 104 to the forwarding entry in table 106).

In the prior art of Fig. IA, a recursively resolved route, e.g., route 108 in the routing table 102 (having route prefix 60.60.60, and having next hop 30.30.30.30) is transformed into an entry in forwarding table 104 having a prefix 108A, and a pointer that points to the same forwarding entry 112 that is pointed to by the route 110A (corresponding to route 110 in routing table 102) that resolves the next hop address of route 108. Therefore when a route such as route 110 that is not recursively resolved is updated, it is sufficient to update its forwarding entry 112 to automatically update all the routes whose next-hop resolves over route 110, which in Fig. IA, include the routes 108 and 114 in the routing table 102 (or routes 108A and 114A in forwarding table 104).

Such a change is illustrated in Fig. IB. Specifically, route 110 is modified to resolve over a different link, indicated by 5.5.5.5 on link 5. By updating the forwarding entry 112 for route 110A, the forwarding entries for routes 108A and 114A are automatically updated as well. Thus, the overhead during switchover in case of route changes is reduced since the number of next-hops is proportional to the number of IGP routes (hundreds) while the number of routes is proportional to the number of EGP routes.
However, this prior art implementation has some limitations. For instance, when there are multiple levels of recursion, recursive routes might still need to be re-downloaded to the forwarding table. This is described with reference to Fig. 2A and Fig. 2B. Route 210 resolves over route 212 which resolves over route 214, pointing to forwarding entry 216 (210→212→214→216), then the forwarding entry pointers for routes 210, 212, and 214 would all point to the link-layer forwarding entry for route 214, which is forwarding entry 216. In the event that route 212 is changed to resolve over route 218, which points to forwarding entry 220, then the route 212 pointer must be updated to point to 220, but in addition, 210 has to be updated to also point to 220 because it used route 214 in the recursive resolution of its next hop. This is shown in Fig. 2B. Thus, when a particular route changes, every other route that used that particular route in a recursive resolution must also have its pointer updated.

Similarly, when the resolving route is deleted or a more specific match for the route next-hop is added, the recursive route resolves over a different route and therefore has to be updated to point to a different forwarding entry. The scenario when a resolving route is deleted is depicted in Figs. 3A and 3B. In Fig. 3A, routes 309A and 310A both have 30.30.30.30 as a next hop, and resolve over route 308A. Note that route 308A has a 24 bit match, which is a better match than route 312A, having only a sixteen bit match. Thus, forwarding table entries 316A and 319A (as well as entry 318A) include pointers to link-layer routing data 322A.

Fig. 3B depicts the routes available after route 308A is removed from the routing table 302A, which also results in the removal of the forwarding entry 322A. Routes 309B and 310B now resolve over route 312B in routing table 302B. Forwarding table prefix entries 314B and 319B have to be updated to point to link layer forwarding entry 320B, associated with the link layer forwarding data for route 312B.

The above examples show specific disadvantages of the prior art. When there are route changes (including route modifications, additions or deletions) to the intermediate routes used to resolve a next hop in a recursively resolved route, then the recursively-resolved routes affected by the change need to be re-resolved, and then that data needs to be translated into forwarding table updates. For recursively resolved routes, a single route change affects all routes resolving over it. Because current Internet routing tables may have on the order of 150 thousand routes, a route change can trigger thousands of updates. The prior art routing table and forwarding table management systems are incapable of efficiently handling the updates.

**SUMMARY**

In exemplary embodiments, the method of routing data packets comprises the steps of: (i) for routes comprising a route prefix and a next hop address, generating a routing
table having a plurality of route records, wherein each route record has a routing prefix entry and a corresponding next hop entry; (ii) for each route in the routing table having a recursively-resolvable next hop, generating a pseudo-route record in the routing table for a pseudo route, wherein the pseudo-route record comprises the corresponding next hop entry and recursively-resolved link-layer routing data for that next hop entry; and, (iii) routing packets using the routing table. Thus, the routing table will contain the routes available to the router (preferably provided by various routing protocols), and will also contain a pseudo route for each next hop address that is not directly resolvable.

In this manner, route updates affecting recursively-resolved routes may be accomplished by simply changing the link-layer routing data of the pseudo route associated with the next hop of the recursively-resolved route.

To facilitate rapid updates in the event of a route change, some embodiments may track which routes are used in recursive resolution. By maintaining a record indicative of the routes used to resolve any given route, the routes affected by a route change may be quickly identified and updated. In one embodiment, each route has a list of all the pseudo routes that utilized the route, thereby enabling quick identification of all pseudo routes requiring re-resolution in the event the route status changes. The record of utilized routes may be maintained using a linked-list. A double-linked list or a circular linked list may also be used.

In other embodiments, the method of routing packets over network routes, where the routes have a routing prefix and a next hop address, comprises the steps of: (i) for directly resolvable routes, generating a forwarding table entry containing link-layer data corresponding to the next hop address, and providing a forwarding table pointer to the forwarding table entry; and (ii) for each recursively-resolved route, generating a pseudo-route forwarding table entry containing link-layer forwarding data corresponding to the recursively-resolved route's next hop address, and providing a forwarding table pointer associated with the recursively-resolved route's routing prefix that points to the corresponding pseudo-route forwarding table entry; and (iii) routing packets using the forwarding table.

In one preferred embodiment, the pointers are hardware independent, and may take the form of a next-hop ID, in which case the next hop IDs are also used as an index into the forwarding table entries. Alternatively, the pointers are memory addresses where the forwarding link data is stored.

In one preferred method, the routing of packets is performed by first matching a destination address of an incoming packet to a routing prefix and obtaining the corresponding forwarding table pointer. Then the corresponding link-layer forwarding data may be used to route the packet.
In another alternative embodiment, the method of routing data packets comprising the steps of: (i) providing a first set of link-layer forwarding entries for directly resolvable routes; (ii) providing a second set of link-layer forwarding entries for recursively-resolvable routes, wherein each forwarding entry in the second set of link-layer forwarding entries is generated based on a recursively-resolved next hop address of at least one recursively-resolvable route; (iii) associating route prefixes for directly-resolvable routes with forwarding entries from the first set of link-layer forwarding entries and associating route prefixes for recursively-resolvable routes with forwarding entries from the second set of link-layer forwarding entries; and, forwarding packets using the link-layer forwarding entries associated with the route prefixes.

Again, the route prefixes may be associated with the first and second sets of link-layer forwarding entries in a hardware independent manner, such as by using next hop identifiers (IDs), or by a hardware dependent method, such as by memory address locations.

The preferred method may use tables to arrange and store the routing information. For example, the first and second sets of forwarding entries may comprise a first table, and a second table may include the route prefix entries and corresponding address entries that point to the appropriate forwarding entry in the first table. In one preferred embodiment, the second table is implemented in a ternary content addressable memory (TCAM) device, and the first table is stored in RAM.

By removing the need to recursively resolve routes on the fly, the routing system described herein provides the ability to retrieve the forwarding entry matching an incoming packet with a minimum number of search operations. In addition, the data structures and management systems described herein provide an efficient update mechanism. This is a particular advantage in embodiments with numerous forwarding engines that utilize the same route and forwarding information, because this enables all of the forwarding tables to be kept in sync with the routing table manager.

These as well as other features, advantages and alternatives will become apparent to those of ordinary skill in the art by reading the following detailed description, with appropriate reference to the accompanying drawings.

**BRIEF DESCRIPTION OF FIGURES**

Figures 1A and 1B are prior art routing tables;
Figures 2A and 2B are prior art routing tables;
Figures 3A and 3B are prior art routing tables;
Figure 4A is a block diagram illustrating a preferred embodiment of the routing system;
Figure 4B is a block diagram illustrating a preferred embodiment of the routing tables;
Figures 5A and 5B conceptually illustrate one example of a route update;
Figures 6A and 6B conceptually illustrate one example of a route update;
Figures 7A and 7B conceptually illustrate one example of a route update; and,
Figures 8 and 9 conceptually illustrate preferred methods of routing packets.

DETAILED DESCRIPTION

I. Routing Table Manager

With respect to Fig. 4A, the data structures and data management systems of one preferred routing system 400 will be described. It should be understood that the router system illustrated in Figure 4A and other arrangements described herein are set forth for purposes of example only, and other arrangements and elements can be used instead and some elements may be omitted altogether, depending on manufacturing and/or consumer preferences.

The Routing Table Manager (RTM) 402 maintains a routing table 404/406 containing routing records. The routes contained in routing records are preferably installed by routing protocols (RP) 408. In one embodiment, the RTM acts as a server, and the RPs 408 behave as clients. The RPs 408 can register routes with the RTM 402, and request to be notified by the RTM 402 if their routes' status changes. In some embodiments, an RP 408 may register a next hop with the RTM 402 before installing a route using that next hop. The RTM 402 may then notify the RP 408 when the next hop is resolvable. Alternatively, the RP 408 may install the route, and await confirmation from the RTM 402 that the route has been resolved. Routes that have next hops that are not resolvable may be put on a pending list, and periodically checked to see if they are resolvable.

In the embodiment of Figure 4A, the RTM 402 provides routing table data to the Forwarding Information Base (FIB) 410. For each prefix in its table, RTM 402 selects as active the best route according to a defined policy and downloads it into the FTB 410. The FIB 410 is the module which manages the Forwarding Table (FT) 412/414 containing the set of downloaded routes.

The FTB 410 preferably customizes the routing data for use by the Forwarding Engine (FE) 416, which actually performs address lookups to obtain the appropriate link layer forwarding data. This architecture allows the RTM 402 to work with routes in a hardware-independent fashion (e.g., utilizing route IDs and next hop IDs), while the FTB 410 can manipulate the data for the particular hardware used by the FE 416 (e.g., using hardware-specific data structures, storage, and addressing, including pointers). On routers handling packets at rates of gigabits per second or more, the FE 416 is usually implemented in hardware. In some embodiments, there are multiple FEs on separate data processing cards with in a chassis, and the RTM 402 downloads the routes to each FIB that manages its local FT.
Those of ordinary skill in the art will recognize that the functions described herein may be performed by alternative embodiments having fewer or more functional entities, and the architectural depiction of Fig. 4A for the RTM 402, FIB 410, and FE 416 is merely one embodiment. In fact, for some embodiments, the FIB 410 and FE 416 are not used.

In table 404, each routing record preferably contains a routing prefix entry 404A and a next hop address entry 404B as shown. The routing prefix 404A preferably specifies a range of IP addresses for which the route is applicable, and the next hop address entry 404B may contain the IP address for the next hop to which the packet should be routed. If the route is directly resolvable, the next hop address entry may alternatively contain or include link layer data, such as a physical or virtual port identifier (e.g., 'Link 3') that allows the routing device to determine the port over which to forward packets for that route. In addition, the link layer data may include other layer 2 data, such as a media access control (MAC) address, a virtual circuit identifier (VCI), a virtual path identifier (VPI), a label that may be pushed before forwarding the packet to an LSP tunnel interface, or any other data needed to route the packets.

Pseudo routes are maintained in routing table 406 (which is preferably an integrated master routing table, reflecting the preferred routes), where each routing entry preferably includes a next hop address field 406A and corresponding link layer data 406B. In one embodiment, the pseudo route entries are preferably generated when a route in table 404 specifies a next hop address that is not directly resolvable. The pseudo route is generated to provide what appears to be a route that may be used to directly resolve the next hop address 404B/406A. In fact, the generation of the pseudo route may involve multiple recursive lookups to determine the link layer data 406B corresponding to the ultimate next hop. The creation and use of pseudo routes in table 406 does not require significant additional memory since the number of next-hops and therefore of pseudo-routes is at most limited to a few hundreds.

The recursive lookups are performed using the routes available to the RTM. In some cases, the RPs 408 may install more than one route specifying the same prefix. In such cases, the RTM will select the best route according to a policy, which may include evaluating preferred routes based on one or more 'cost' metrics (e.g., delay, expense, and error are three such metrics, where the delay cost metric reflects the amount of delay on the link, the expense cost metric reflects the communications cost associated with using the link, and the error cost metric reflects the error rate of the link). Alternatively, routes may be evaluated and selected based on the RP 408 that installed the route, where routes from certain RPs 408 are given priority over those installed by other RPs 408. These preferred routes are then entered into an integrated master routing table.
The pseudo routes may be periodically checked to see if more preferred routes may be used to recursively resolve the next hop addresses. For instance, an RP 408 may install a route having a longer prefix match than one used to resolve a next hop. The resolved pseudo routes may be maintained in a list, such as a linked list, to facilitate route update and/or route verification management.

The routing tables in Figure 4B demonstrate the use of pseudo routes. The first three routes in table 450 specify next hop data 452, 454, 456 corresponding to IP addresses that are not directly resolvable. Therefore, corresponding pseudo routes 466, 468 and 470 are created in table 462. Note that next hop data 458 and 460 also use those pseudo routes. To facilitate route updates, the pseudo routes 466, 468, and 470 have pointers as shown by lines 464, 472, and 474, respectively, which identify the routes used to resolve the pseudo routes. Note that routes 466 and 468 both use route 467, and the three routes are linked in a circular double-linked list indicated by pointers 472, 474, and 465.

In alternative embodiments, directly resolvable routes may be treated in a superficially similar manner to allow for uniform handling and routing-record management. In particular, all routes, both directly-resolvable and recursively resolvable, may be assigned a next hop identifier (next hop ID). The next hop ID may then be used as next hop data 404B in table 404. Then, a record may be generated in table 406 for each next hop ID, and stored in data fields 406A and 406B. For directly resolvable routes, the corresponding link layer data 406B is simply entered for the corresponding next hop ID 406A. For recursively-resolvable routes, the RTM 402 generates link layer data 406B using recursive resolution.

In either embodiment, the next hop entry (e.g., next hop IP address, next hop link layer data, next hop ID) preferably corresponds to a single IP address. The pseudo route table 406, therefore, contains link layer data for the specific next hop data. In this way, recursively-resolved routes each have their own corresponding pseudo route entry that contains the appropriate link layer data. Of course, if two routes specify the same next hop IP address (e.g., 456, 458, or 454, 460), they may share the pseudo route for that next hop address.

The routing tables including pseudo routes described above are updated as shown in the example of Figs. 5A and 5B. In Fig. 5A, next hop entries 508A and 510A both specify recursively-resolvable IP addresses 30.30.30.30. A pseudo route entry 516A is created for the next hop 30.30.30.30. Pseudo route 516A is resolved using route 30.30.30.0/24 and the forwarding data 512A is used as link layer data entry 513A. The routing tables 502B/503B may then be used to route packets.

In some preferred embodiments, the data in tables 502B/503B is used to generate tables 504A and 506A. Specifically, pseudo route 'prefix' 522A points to link layer
forwarding entry 528A, which was obtained from entry 513A. It should be understood 
that in most embodiments, entry 522A need not actually be loaded into table 504A 
with a pointer to forwarding entry 528A. This is because the entries in table 504A are 
preferably only those that need to be looked up for routing purposes. In most cases, 
incoming packets will not be addressed to the IP addresses of the pseudo routes, and 
hence the pseudo routes do not need to be made available for matching purposes. 
Route prefixes 518A and 520A also point to pseudo route forwarding entry 528A.

In Fig. 5B, link layer data 512B has been modified. Pseudo route 516B is identified 
as having used the route that was modified, and the RTM re-resolves the pseudo route 
516B. The link layer data 513B is then updated to contain 5.5.5.5 Link 5.

In the forwarding tables 504B/506B, the routes requiring updates are the pseudo 
route 552B/528B, which is updated to have 528B changed to 5.5.5.5 Link 5, and the 
link-layer data 526B. Thus, due to the above-described pseudo route data structure, 
route updates affecting recursively-resolved routes may be accomplished by simply 
changing the link-layer routing data of the pseudo route associated with the next hop of 
the recursively-resolved route.

To facilitate rapid updates in the event of a route change, some embodiments may 
track which routes are used in recursive resolution. By maintaining a record indicative 
of the routes used to resolve any given route, the routes affected by a route change may 
be quickly identified and updated. In one embodiment, each route has a list of all the 
pseudo routes that utilized the route, thereby enabling quick identification of all pseudo 
routes requiring re-resolution in the event the route status changes. The record of 
utilized routes may be maintained using a linked-list. A double-linked list or a circular 
linked list may also be used. As described above, the pointer indicated by 514A/514B 
(preferably part of a linked list, double linked list, or circular linked list) was used to 
identify pseudo route 516B as requiring re-resolution. Note that in Figs. 6A, 6B, 7A 
and 7B, lines indicating the linked lists are omitted.

Another example of a route update is shown in Figs. 6A and 6B. In Fig. 6A, 
routing table 602A includes routes 610A, 611A, 612A, and 614A that have next hops 
that are not directly resolvable. Pseudo routes 616A, 618A, and 619A are generated in 
table 608A to provide link layer routing data for next hops 20.20.20.20, 30.30.30.30, 
and 50.50.50.50, respectively. Note that a double recursion is required to resolve routes 
610A and 614A. In a preferred embodiment, the pseudo routes 616A, 618A, and 619A 
are loaded into the forwarding tables 604A, 606A, where table 604A contains route 
prefixes and pointers to forwarding entries, and table 606A contains the link-layer 
forwarding data at the correct address, or index, of the table. The link-layer data from 
table 608A is entered into forwarding table 606A as forwarding entry 620A for pseudo 
route 20.20.20.20 628A, while link-layer data 622A is entered for the forwarding entry
for pseudo route 30.30.30.30 618A, and link-layer data 628A is entered for the forwarding entry for pseudo route 50.50.50.50 619A.

In Fig. 6B, the next hop for route 612A is modified to be 50.50.50.50 as shown in field 612B. This route is an intermediate route used in the recursive resolution of routes 610B and 614B because it was used in the resolution of the next hop address 30.30.30.30. As a preliminary matter, the route change altering data 612B is updated in forwarding tables 604B and 606B by changing the pointer from prefix 625B to point to the forwarding entry for the pseudo route for next hop 50.50.50.50 628B. Note also that because the pseudo route 20.20.20.20 is no longer used, it may be deleted from tables 604B, 606B, 608B.

The route updates for routes indirectly affected by this change (routes 610B and 614B) are performed as follows: for the routing table update, pseudo route 30.30.30.30 616B is re-resolved, and new link-layer forwarding data 609B is obtained from recursively resolving the route next hop address, and in particular, from route 610B; for the forwarding table update, all that needs to be done to update routes 624B and 626B is to update forwarding entry 622B to be 5.5.5.5 link 5. In each case, only a single forwarding entry needs to be adjusted to accommodate changes to all the routes using that next hop, and no pointers need to be adjusted. In the prior art described above, for all routes using next hop 30.30.30.30, either all the pointers or all the forwarding entries would need to be updated.

In the example of route updates in Figs. 7A and 7B, a route is deleted, which affects the routes that used the deleted to route to recursively resolve their next hops. In Fig. 7A, routes 710A and 712A of table 702A specify a next hop of 30.30.30.30. A pseudo route entry is generated for next hop 30.30.30.30, and entered into table 708A as link-layer forwarding data 709A 3.3.3.3 Link 3. In the forwarding tables 704A, 706A, the forwarding table prefixes 724A 40.40.40.0/16 and 726A 60.60.60.0/24 both have pointers to the link layer forwarding data 720A for the pseudo route 30.30.30.30.

In Fig. 7B, when the route 30.30.30.0/24 is no longer present, the pseudo route 30.30.30.30 must be re-resolved using route 711B 30.30.0.0/16. The pseudo route link layer forwarding data entry 709B in the routing table 708B is updated accordingly to 5.5.5.5 Link 5. In the forwarding tables 704B, 706B, the only change that needs to be made is the link-layer routing data 722B for pseudo route 30.30.30.30.

Thus, in the RTM described herein, one preferred method 800 of routing data packets is shown in Fig. 8. At step 802, compiling, or generating, a routing table. The routes in the routing table each comprise a route prefix and a next hop address, and so the data records in the routing table each has a routing prefix entry and a corresponding next hop entry. At step 802, pseudo-route records are generated at least for each route in the routing table having a recursively-resolvable next hop. The pseudo-route record
comprises the corresponding next hop entry and recursively-resolved link-layer routing data for that next hop entry. At step 806, the packets are routed using the routing table. Of course, as described above, the method 800 includes various ways of storing the routing data. In particular, various pointers or other software constructs may be used, as well as hardware memory devices. Preferably, the pointers in method 800 are hardware independent, and may take the form of a next-hop ID.

The method 800 may also generating pseudo routes even for routes that are directly resolvable, which allows for similar data structures to be used for recursive and non-recursive routes. Thus, the routing table will contain the routes available to the router (preferably provided by various routing protocols), and will also contain at least one pseudo route for each next hop address that is not directly resolvable.

In another embodiment shown in Fig. 9, the method 900 of routing packets over network routes is described. Again, the routes have a routing prefix and a next hop address. Method 900 begins at step 902, where for each directly resolvable route, forwarding table entries containing link-layer data corresponding to the next hop address are generated, and a forwarding table pointer to the forwarding table entry is provided. At step 904, for each recursively-resolved route, a pseudo-route forwarding table entry containing link-layer forwarding data corresponding to the recursively-resolved route's next hop address are generated, and a forwarding table pointer associated with the recursively-resolved route's routing prefix that points to the corresponding pseudo-route forwarding table entry is provided. Finally, at step 906, the packets are routed using the forwarding table.

In another alternative embodiment, the method of routing data packets comprises the steps of: (i) providing a first set of link-layer forwarding entries for directly resolvable routes; (ii) providing a second set of link-layer forwarding entries for recursively-resolvable routes, wherein each forwarding entry in the second set of link-layer forwarding entries is generated based on a recursively-resolved next hop address of at least one recursively-resolvable route; (iii) associating route prefixes for directly-resolvable routes with forwarding entries from the first set of link-layer forwarding entries and associating route prefixes for recursively-resolvable routes with forwarding entries from the second set of link-layer forwarding entries; and, forwarding packets using the link-layer forwarding entries associated with the route prefixes.

Again, the route prefixes may be associated with the first and second sets of link-layer forwarding entries in a hardware independent manner, such as by using next hop identifiers (IDs), or by a hardware dependent method, such as by memory address locations.

Preferably, the routing of packets using forwarding tables is performed by first matching a destination address of an incoming packet to a routing prefix and obtaining
the corresponding forwarding table pointer. Then the corresponding link-layer forwarding data may be used to route the packet. In one preferred embodiment, the pointers are hardware independent. Alternatively, the pointers are memory addresses where the forwarding link data is stored. For example, the first and second sets of forwarding entries may comprise a first table, and a second table may include the route prefix entries and corresponding address entries that point to the appropriate forwarding entry in the first table. The second table may be implemented in a ternary content addressable memory (TCAM) device. That is, the prefixes are stored in a content addressable memory such that it performs a longest prefix match with the destination address of the packet to be routed. The best match within the TCAM will result in a pointer value. The pointer value is an address of a forwarding entry in the first table, where the first table may be stored in a more conventional RAM memory device, such as SRAM, DRAM, or other directly addressable memory device.

It should be understood that the processes, methods and networks described herein are not related or limited to any particular type of software or hardware, unless indicated otherwise. For example, operations of the packet switch may be performed through application software, hardware, or both hardware and software. In view of the wide variety of embodiments to which the principles of the present embodiments can be applied, it is intended that the foregoing detailed description be regarded as illustrative rather than limiting, and it is intended to be understood that the following claims including all equivalents define the scope of the invention.
Claims

[1] A method of routing packets over network routes, the routes having a routing prefix and a next hop address, comprising:
   a) for each directly resolvable route:
      (i) generating a forwarding table entry containing link-layer data corresponding to the directly-resolvable route's next hop address;
      (ii) providing a forwarding table pointer associated with the directly-resolvable route's routing prefix that points to the corresponding generated forwarding table entry;
   b) for each recursively-resolved route:
      (i) generating a pseudo-route forwarding table entry containing link-layer forwarding data corresponding to the recursively-resolved route's next hop address;
      (ii) providing a forwarding table pointer associated with the recursively-resolved route's routing prefix that points to the corresponding pseudo-route forwarding table entry; and
   c) routing packets using the forwarding table.

[2] The method of claim 1 wherein the pointers are hardware independent next-hop IDs, and the next hop IDs are also used as an index into the forwarding table entries.

[3] The method of claim 1 wherein the pointers are memory addresses where the forwarding link data is stored.

[4] The method of claim 1 wherein the step of routing packets comprises the steps of:
   matching a destination address to a routing prefix;
   obtaining the corresponding forwarding table pointer;
   using the corresponding link-layer forwarding data to route the packet.

[5] The method of claim 1 further comprising the step of maintaining a record of utilized routes indicative of the routes used in the recursive resolution of a recursively-resolved route.

[6] The method of claim 5 wherein the record of utilized routes is maintained using a linked-list.

[7] The method of claim 5 further comprising:
   determining a change in status of a route in the record of utilized routes; and,
   updating the pseudo-route forwarding table entries of affected recursively-resolved routes.

[8] A method of routing data packets comprising the steps of:
for routes comprising a route prefix and a next hop address, generating a routing
table having a plurality of route records, wherein each route record has a routing
prefix entry and a corresponding next hop entry;
for each route in the routing table having a recursively-resolvable next hop,
generating a pseudo-route record in the routing table for a pseudo route, wherein
the pseudo-route record comprises the corresponding next hop entry and re-
cursively-resolved link-layer routing data for that next hop entry; and,
routing packets using the routing table.

[9] The method of claim 8, further comprising the step:
for at least one route in the routing table, maintaining a list of all pseudo routes
that utilized the at least one route in order to recursively resolve the link-layer
routing data for the pseudo route.

[10] The method of claim 9 wherein the list of all pseudo routes is a double-linked
list.

[11] The method of claim 1 wherein the link layer data comprises a MAC address.

[12] A method of routing data packets comprising the steps of:
providing a first set of link-layer forwarding entries for directly resolvable
routes;
providing a second set of link-layer forwarding entries for recursively-resolvable
routes, wherein each forwarding entry in the second set of link-layer forwarding
entries is generated based on a recursively-resolved next hop address of at least
one recursively-resolvable route;
associating route prefixes for directly-resolvable routes with forwarding entries
from the first set of link-layer forwarding entries;
associating route prefixes for recursively-resolvable routes with forwarding
entries from the second set of link-layer forwarding entries;
forwarding packets using the link-layer forwarding entries associated with the
route prefixes.

[13] The method of claim 12 wherein the method of associating route prefixes with
the first and second sets of link-layer forwarding entries is hardware in-
dependent.

[14] The method of claim 13 wherein the method of associating route prefixes with
the first and second sets of link-layer forwarding entries utilizes next hop IDs.

[15] The method of claim 12 wherein the method of associating route prefixes with
the first and second sets of link-layer forwarding entries is hardware dependent.

[16] The method of claim 12 wherein the first and second sets of forwarding entries
comprise a first table, and the method of associating route prefixes with the first
and second sets of link-layer forwarding entries comprises a second table
containing routing records, each routing record comprising a route prefix entry and an address entry that points to a forwarding entry in the first table.

[17] The method of claim 16 wherein the first and second tables are implemented in a ternary content addressable memory device.
### ROUTING TABLE

<table>
<thead>
<tr>
<th>DESTINATION</th>
<th>NEXT-HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.0/16</td>
<td>50.50.50.50</td>
</tr>
<tr>
<td>20.20.0.0/16</td>
<td>30.30.31.31</td>
</tr>
<tr>
<td>40.40.0.0/16</td>
<td>30.30.30.30</td>
</tr>
<tr>
<td>30.30.30.0/23</td>
<td>3.3.3.3 ON LINK 3</td>
</tr>
<tr>
<td>3.3.3.0/24</td>
<td>LINK 3</td>
</tr>
<tr>
<td>4.4.4.0/24</td>
<td>LINK 4</td>
</tr>
<tr>
<td>5.5.5.0/24</td>
<td>LINK 5</td>
</tr>
<tr>
<td>50.50.50.0/24</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
<tr>
<td>60.60.60.0/24</td>
<td>30.30.30.30</td>
</tr>
<tr>
<td>70.70.70.0/24</td>
<td>30.30.31.31</td>
</tr>
</tbody>
</table>

### PSEUDO-ROUTING TABLE

<table>
<thead>
<tr>
<th>PSEUDO ROUTE</th>
<th>ULTIMATE NEXT-HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.30.30</td>
<td>3.3.3.3 ON LINK 3</td>
</tr>
<tr>
<td>30.30.31.31</td>
<td>3.3.3.3 ON LINK 3</td>
</tr>
<tr>
<td>50.50.50.50</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
</tbody>
</table>

**FIG. 4B**
FIG. 6B

**ROUTING TABLE**

<table>
<thead>
<tr>
<th>DESTINATION</th>
<th>NEXT-HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.40.0.0/16</td>
<td>30.30.30.30</td>
</tr>
<tr>
<td>3.3.3.0/24</td>
<td>50.50.50.50</td>
</tr>
<tr>
<td>5.5.5.0/24</td>
<td>LINK 5</td>
</tr>
<tr>
<td>20.20.20.0/24</td>
<td>3.3.3.3 ON LINK 3</td>
</tr>
<tr>
<td>30.30.30.0/24</td>
<td>50.50.50.50</td>
</tr>
<tr>
<td>50.50.50.0/24</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
<tr>
<td>60.60.60.0/24</td>
<td>30.30.30.30</td>
</tr>
</tbody>
</table>

**PSUEDO-ROUTING TABLE**

<table>
<thead>
<tr>
<th>PSEUDO ROUTE</th>
<th>ULTIMATE NEXT-HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.30.30</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
<tr>
<td>50.50.50.50</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
</tbody>
</table>

**FORWARDING TABLE PREFIXES**

<table>
<thead>
<tr>
<th>PREFIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.40.0.0/16</td>
</tr>
<tr>
<td>3.3.3.0/24</td>
</tr>
<tr>
<td>5.5.5.0/24</td>
</tr>
<tr>
<td>20.20.20.0/24</td>
</tr>
<tr>
<td>30.30.30.0/24</td>
</tr>
<tr>
<td>50.50.50.0/24</td>
</tr>
<tr>
<td>60.60.60.0/24</td>
</tr>
</tbody>
</table>

**FORWARDING ENTRIES**

<table>
<thead>
<tr>
<th>ENTRIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINK 5</td>
</tr>
<tr>
<td>3.3.3.3 ON LINK 3</td>
</tr>
<tr>
<td>5.5.5.5 ON LINK 5</td>
</tr>
</tbody>
</table>

**PSEUODO ROUTING**

<table>
<thead>
<tr>
<th>PSEUODO ROUTE</th>
<th>ULTIMATE NEXT-HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.30.30</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
<tr>
<td>50.50.50.50</td>
<td>5.5.5.5 ON LINK 5</td>
</tr>
<tr>
<td>Forwarding Entries</td>
<td>5.5.5.5 Link 5</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Forwarding Prefixes</td>
<td>30.30.0.0/16</td>
</tr>
<tr>
<td></td>
<td>60.60.60.0/24</td>
</tr>
</tbody>
</table>

**Routing Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>5.5.5.5 on Link 5</th>
<th>30.30.30.30</th>
<th>5.5.5.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.0.0/16</td>
<td>40.40.0.0/16</td>
<td>3.3.3.0/24</td>
<td>60.60.60.0/24</td>
<td></td>
</tr>
</tbody>
</table>

**Pic. 7B**

**Pseudo-Routing Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Ultimate Next-Hop</th>
<th>5.5.5.5 on Link 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.30.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Pseudo Route**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Ultimate Next-Hop</th>
<th>5.5.5.5 on Link 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.30.30.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>