Fast IP Routing Lookups and Updates in Hardware

Cheng Chi You, Yuan Sun Chu, Chi Fang Li, Hui Kai Su

Computer Network Laboratory, Department of Electrical Engineering,

National Chung Cheng University, Taiwan, R.O.C

m8831@cn.ee.ccu.edu.tw pat@cn.ee.ccu.edu.tw

Abstract

The rapid growth of the Internet has created a demand for the router that can process heavy traffic and achieve gigabit speed. Currently, the process of IP lookups is done in software and has become a major performance bottleneck for the router. In connection with this problem, many fast IP routing lookup mechanisms are proposed. But most of them focus on lookups. In this paper, we propose a novel structure of fast lookups and updates in hardware. Implemented in a pipelined fashion, the lookup speed can achieve one memory access for every IP lookup.

Ⅰ. Introduction

Statistics show the network traffic on the Internet is doubling each few months. In order to provide nice quality-of-service (QoS), three-key design issues on the next generation IP routers are link speeds, router throughput, and packet forwarding rates. The first two issues are now readily available. For example, fiber-optic cables can provide faster link speeds, and new IP switching technology (Layer-3 switching or multilayer switching) can be used to transmit packets from the input interface of the route to the corresponding output interface at gigabit rates. This paper deals with the third design issue, packet forwarding, which performs poorly as link speeds increase at present.

Since the Internet grows exponentially, some problems occur in IP address space. The IP address space is divided into classes A, B, and C (allowing sites to have 24, 16, and 8 bits respectively for addressing). Nevertheless, it is too inflexible and wasteful of IP address space, especially the class B address. To make better use of the scarce resource and reduce IP routing table entries, Classless Inter-Domain Routing (CIDR) is presented in 1993 [1]. With CIDR, each route is identified by a <prefix, prefix length> pair, that the prefix length is between 0 and 32 bits. If a coming destination IP address matches many route prefixes, the longest prefix length will be selected. This is called longest matching prefix. For example, there are two routes, <140.123/16>, and <140.123.107/24>, in the IP routing table. The IP address 140.123.107.64 will select the second entry.

In recent years, a number of methods have been proposed for improving the performance of longest matching prefix search [2, 3, 4, 5, 6, 7, 8, and 9]. Most of them can achieve high average search throughput, but are slow in the updating speed.

In this paper, Section Ⅱ describes the lookup scheme. Section Ⅲ introduces the update and the insertion . Section Ⅳ introduces the deletion. Section Ⅴ discusses how to improve the update and insertion. Section Ⅵ is the conclusion.

This work is in part supported by NSC, under the grant NSC 89-2215-E-194-009.

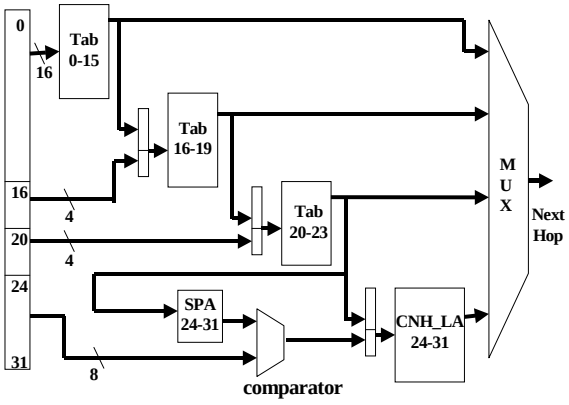

Figure 1. Architecture for the proposed IP routing lookup mechanism

III. Proposed Scheme

The architecture for the proposed scheme is shown in Figure 1. The architecture is divided to four levels. The first level is composed of Tab0_15, which stores all possible routes that are shorter than 17 bits. This table has 2^16 entries, addressed from 0.0 to 255.255. For example,

the route 50/8 has 2^(16-8)=256 entries associated in Tab0_15. The 256 entries have the same next hop.

The second level is composed of Tab16_19, which stores all possible routes that are longer than 16 bits and shorter than 21 bits. Each 16-bit prefix that has at least one route longer than 16 bits is allocated 2^4=16 entries in Tab16_19. For example, assume that there are already two routes (50/8, 50.123.240/20) in the table. Next hop distributions are shown in Figure 2. The first route requires entries addressed from 50.0 to 50.255 (except for 50.123) in Tab0_15. The second route requires Tab16_19 to be used. So, in Tab0_15, the index (5) is entered into the entry corresponding to the 50.123 prefix. In Tab16_19, 16 entries starting with memory location 5*16 are allocated. Most of these locations are full of the next hop corresponding to the route 50/8, but one of them (5*16+15) is full of the next hop corresponding to the route 50.123.107/20.

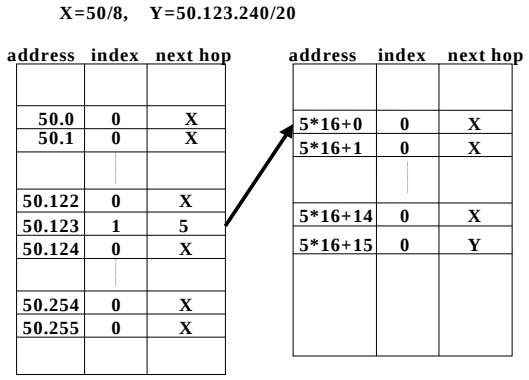

Figure 2. Example of two routes in Tab0_15 and Tab16_19.

The third level is composed of Tab20_23, which stores all possible routes that are longer than 20 bits and shorter than 25 bits. Each 20-bit prefix that has at least one route longer than 20 bits is allocated 2^4=16 entries in Tab20_23.

The fourth level is composed of SPA and CNH_LA. They store all possible routes that are longer than 24 bits. Each 24-bit prefix that has at least one route longer than 24 bits is allocated 64 bits in SPA and 8 entries in CNH_LA. SPA/CNH_LA is a kind of technique that the memory size is compressed. For example, assume IP address is 8-bit long and there are four routes (0/1, 010/3, 001001/6, 010101/6) in the table. Next hop distributions are shown Figure 3. We will use SPA/CNH_LA architecture to establish the table. The result is shown in Figure 4. SPA stores all starting points corresponding to next hops shown Figure 3. CNH_LA stores all next hops shown Figure 3 and prefix lengths corresponding to routes. Now assume that a packet with the destination address 45 is looked up. Because 45 is between 40 and 64 in SPA, we get an index 2 ($0 \leqq$ address<36 => index 0, $36 \leqq$ address<40 => index1, $40 \leqq$address<64 => index 2, etc). Then we use the index into CNH_LA and find that the next hop is A.

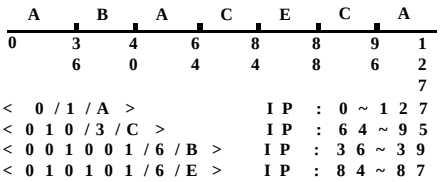

Figure 3. Next hop distributions for four routes.

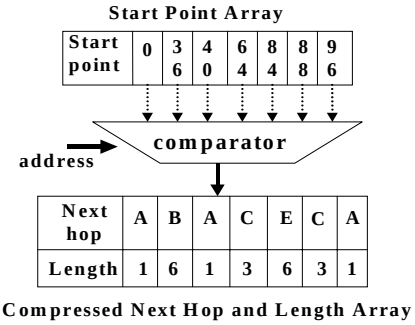

Figure 4. Example of the SPA/CNH_LA architecture with four routes.

When a destination IP address V is looked up, the completely routing lookup mechanism is performed as following.

```
If (Tab0_15[0~15 bits of V].index != 1) {
    Next_hop=Tab0_15[0~15 bits of V].next_hop ;
}
else {
    pointer2=Tab0_15[0~15 bits of V].next_hop*16+
                16~19 bits of V ;
    if (Tab16_19[pointer2].index != 1) {
        Next_hop=Tab16_19[pointer2].next_hop ;
     }
    else {
        pointer3=Tab16_19[pointer2].next_hop*16+
                20~23 bits of V ;
        if (Tab20_23[pointer3].index != 1) {
            Next_hop=Tab20_23[pointer3].next_hop ;
        }
        else {
```

```
            pointer4= Tab20_23[pointer3].next_hop ;
            cnh_la=comparator(SPA[pointer4],
                              24~31 bits of V) ;
            Next_hop=CNH_LA[cnh_la].next_hop ;
        }
    }
}
```

Ⅳ. Update and Insertion

In order to conform to the longest matching prefix, the prefix length for each route becomes the keyword. When a new route with the prefix length L and the updating region R is inserted, we will take the following steps.

1. For each entry in updating region R in the table whose length filed is less than or equal to L, the new next hop is entered.
2. Otherwise, those entries with length filed greater than L are unchanged.

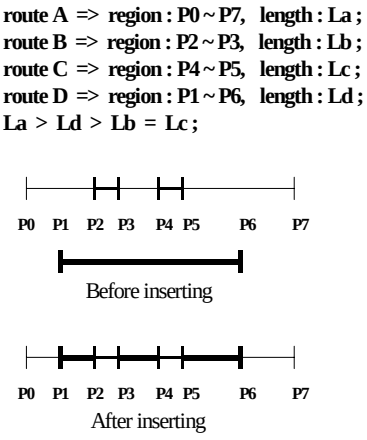

Figure 5. Example for explaining the updating process with SPA/CNH_LA.

Above rules are suitable for the first 3 levels. The process of the update and insertion for SPA/CNH_LA is more complex. If the value of starting point of the updating region or its end point plus one is unequal to any value in SPA, some data in SPA and CNH_LA will be shifted. Figure 5 explains this case. There are three routes (route A, route B, route C) in the table and a new route D is inserted. Because the value of the starting point of the updating region for the route D and its end point plus one are unequal to any points in the table, we must enter some data in the region P1~P2 and P5~P6. Then, we use the prefix length of the route D to judge that weather every entry in region P2~P5 is updated or not.

For example, there are four routes in SPA/CNH_LA in Figure 4, and a route <001/3> will be inserted. Because the value of the starting point of the new route is unequal to any value in SPA, a new data (start point 32, next hop D, length 3) between the 0th entry and the 1st entry in SPA/CNH_LA must be entered. Then the prefix length of the new route is used to update the 1st and 2nd entries in CNH_LA. The final result is shown in Figure 6.

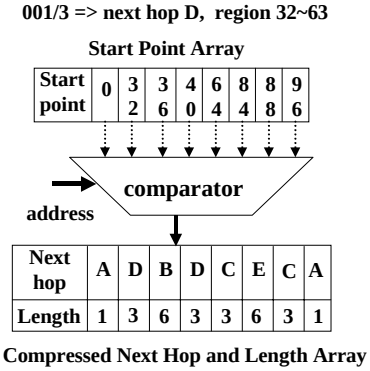

Figure 6. The result after inserting the route <001/3> in SPA/CNH_LA.

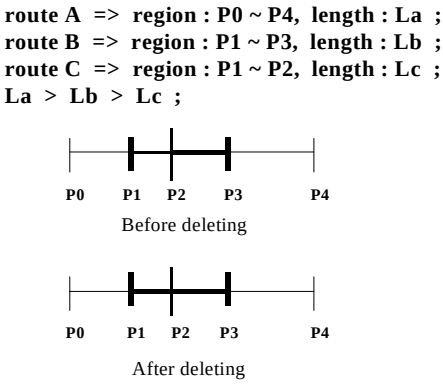

Figure 7. Example for explaining the deleting process.

Ⅴ. Deletion

In deletion, we will search a specific route whose region covers the region of the deleted route and length is closest to the length of deleted route. The specific route may be in the left or the right of the deleted region. Then, the searched route is used to update the deleted region. Figure 7 explains the above descriptions. There are three routes (route A, route B, and route C) in the table and the route C is deleted. Firstly, the entry (P0~P1) is read in the left of the deleted region and its content (the starting point

filed and the length field) is used to judge that this entry belongs to the route A. The route A covers the route C. Secondly, the entry (P2~P3) is read in the right of the deleted region and its content is used to judge that this entry belongs to the route B. The route B also covers the route C. Because the prefix length of the route B is closest to the route C, we use the route C to update the deleted region.

Ⅵ. Improvement for the Update and Insertion

Because our architecture is hierarchy (four levels), updating or inserting may update the table from the first level to the second level (even the four level). We hope that the processes of update or insertion stop in local level and not enter to the next level. Figure 8 explains our methods. There are three routes (route A, route B, and route C) in the table. The route A is established in the first level. The route B and route C are established in the second level. The route A covers the route B and route C. If the route A is updated before improving, we must modify some entries in the first level and second level (region P1~P2, region S0~S1, region S2~S3, and region S4~S5). To achieve our hope, we add a RP register in the table and one RP bit in every entry of the second level. The RP register stores the next hop and prefix length of the route A. The RP bits in regions S0~S1, S2~S3, and S4~S5 are set 1, and others are set 0. When the route A is updated, we just modify the region of the route A in the first level and the RP register.

After adding the RP register, the process for the IP lookups is changed. When an IP address is looked up in the region S0~S1, S2~S3, or S4~S5, the RP bit is set 1 and it represents that the next hop is stored in the RP register.

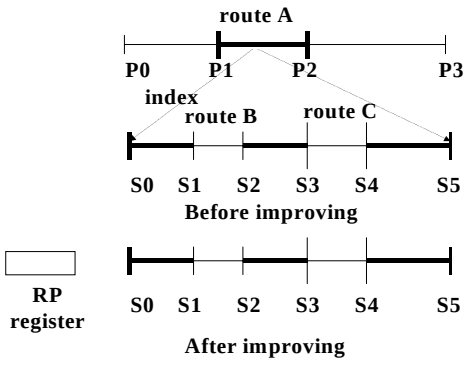

Figure 8. Using the RP register to improve the update and insertion.

Ⅶ. Conclusions

In this paper, we propose a novel architecture with the fast speed for lookups and updates.

In Figure 1, the longest delay time for IP lookups is 5 memory accesses (Tab0_15 → Tab16_19 → Tab20_23 → SPA → CNH_LA). Implemented in a pipelined fashion, the lookup speed can achieve one memory access for every IP lookup.

The memory requirement for our design is small. The large routing table with 20000 routes in Mac-east [10] can be compacted to a forwarding table with about 570 kbytes.

References

1. Y. Rekhter, T. Li. "An Architecture for IP Address Allocation with CIDR" RFC 1518, Sept 1993.
2. S. Nilsson, and G. Karlsson "IP-Address Lookup Using LC-Tries." IEEE Journal on Selected Areas in Communications, Vol. 17, No.6, June 1999, pp. 1083 –1092.
3. M. Waldvogel, G. Varghese, J. Turner, and B. Plattner "Scalable High Speed IP Routing Lookups." in Proc. ACM SIGCOMM'97, France, pp. 25-36.
4. W. Doeringer, G. Karjoth, and M. Nassehi, "Routing on Longest-Matching Prefixes." IEEE/ACM Trans. Networking, vol. 4, Feb. 1996, pp. 86-97.
5. B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using Multiway and Multicolumn Search." IEEE INFOCOM'98, San Francisco, April 1998, Session 10B-2.
6. P. Gupta, S. Lin, and N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds." IEEE INFOCOM'98, San Francisco, April 1998, Session 10B-1.
7. M. Degermark, A. Brodnik, S. Carlsson, and S. Pink "Small Forwarding Table for Fast Routing Lookups." in Proc. ACM SIGCOMM'97, France, pp. 34.
8. N. Huang, and S. Zhao "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers." IEEE Journal on Selected Areas in Communications, Vol. 17, No. 6, June 1999, pp. 324-334.
9. A. McAuley, and P. Francis. "Fast Routing Table Lookup Using CAMs". Proc. IEEE INFOCOM 1993, Vol. 3, pp1382-1391, San Francisco, USA.
10. Merit Networks, Inc. See http://www.merit.edu.