A Design and Implementation of a High Performance IPv6 Lookup Algorithm Based on Hash and Cam

Wang Ruiqing
Department of computer science
XUPT
Xi'an, China

Du Huimin, Wang Yagang
School of electronic engineering
XUPT
Xi'an , China

***Abstract*-With the development of IPv6 networks, IP address lookup has been the bottleneck for high performance routers. In this paper, a two-level IPv6 lookup algorithm based on Hash and CAM is proposed on an analysis on the prefix length distribution of routing table and the growth trend of routing table entries. The prefixes whose length can be divided by 8 are stored in seven Hash RAMs in first level, the remaining prefixes, with their length cannot be divided by 8, will be stored into expanded RAMs in second level. Our proposed scheme was implemented with 8-stage pipeline design and was verified on Xilinx V6- board. The experiments show that the algorithm has high efficient storage utilization, searching rate and updating rate. Furthermore, it is easy to be scalded in hardware.**

Keywords-IPv6, route lookup, two level, Hash, CAM, high performance

I. INTRODUCTION

An IP router[1] connects multiple small networks into a large network and communicates with other routers through IP protocols[2]. It plays a role of bridge on the Internet. IP routers affect the performance of a network significantly. With the rapid growth of the traffic of the Internet, the current IP routers can't satisfy the requirements of the growth. Therefore, the high performance IP routers are expected to come into markets.

The efficiency of IP address lookup[3] determines the performance of an IP router. CIDR (Classless Inter-Domain Routing) addressing scheme of the IP address has been proposed to solve the rapid growth of route information, and the problem of IP address lookup has converted exact matching IP address into the best matching IP address. As a result, when a datagram enters an IP router, it finds entries agreed with the incoming datagram's destination address, selects an entry with the longest prefix and determines the next hop of the datagram, which is called LPM (longest prefix matching) algorithm. Because LPM increases the complexity of IP address lookup and a lot of IPv4 lookup algorithms can not be applied to IPv6 directly, efficiency of lookup algorithms for IPV6 has become one of the popular problems in the field of IP routers. The updating overheads of the lookup table and the lookup speed, scalability, memory utilization of an algorithm are crucial factors to improve the performance of an IP router.

Through analyzing the prefix distributions, a two-level high performance address lookup algorithm for IPv6 was proposed. The algorithms adapts two-level lookup structure based on Hash[4] and CAM[5,6]. An implementation with a 8-stage pipeline has been tested on Xilinx ML605 board with 573Mhz and the results show that approximately 93% of the prefixes can matching in the first level lookup, only minority needs second-level matching. The average rate of lookup is 1 packet/cycle. The average number of memory access is 1.07/lookup and the number is 2 in the worst condition.

The rest of this paper is organized as follows. Section 2 introduces the related lookup algorithms. Section 3 summarizes the prefix distributions and introduces our algorithm. Section 4 simulates and evaluates the proposed algorithm. Finally, some conclusions are given and the further research directions are proposed.

II. RELATED WORK

The current decades have witnessed the developments of the researches on IPv6 routing lookup, here is a classification for a number of the previous IP address lookup.

Reference [7] increase parallalism on Trie structure, which is the most popular data structure for IPv4 address lookup, and achieves a higher throughput. [8] combines the bitmap compression with path compression and employs a variable-stride mechanism to maximize the compress ratio and minimize average memory reference.

[9] is composed of Hash and CAM, in which the simple Hash hardware is used to reduce the lookup time, and the CAM is used to solve the collision problem effectively.

[10] divides an IPv6 address into 8 chunks of 16 bits each and the chunks are reduced recursively through several tunable phases according to the tradeoff between lookup performance and memory consumption.

[11] implement TCAM with 512 bits of prefix entry with 512 bits of destination information, allowing it to implement large address lookups as well as quality of service mechanisms.

III. LOOKUP ALGORITHM FOR IPV6 BASED ON HASH AND CAM

A. Prefix distributions and growth trends for IPv6

The number of the routing table for IPv6 has been increasing and the increasing routing table shows some certain distributions on the prefixes entries from Potaroo[12], as shown in table I. We find the following some interesting features:

Published by Atlantis Press, Paris, France.

①The length of prefixes are completely smaller than 64;

②The ratio of prefixes to the whole lookup table whose length can be divided by 8 is very large, and more than 93%.

③The growth of prefixes that can be divided by 8 is rapid. Moreover, the ratio of prefixes to the whole IPv6 route table growing. For example, prefixes that length is 32 are increasing from 600 to 1176, length of 64 from 10 to more than 30.

④The ratio of prefixes that cannot be divided by 8 is as small as 6%, and grows slowly.

B. Related definitions

Definition 3-1 HP(i)(Hash Prefixes): HP(i) is a set of prefixes whose length can be divided by integer 8 and $i \in [8,16,24,32,40,48,56,64]$.

In this paper, Hash prefixes HP(i) could be HP(16)、HP(24)、HP(32)、HP(40)、HP(48)、HP(56) and HP (64).

Definition 3-2 EP(j, k)(Expanded Prefix): EP(j, k) is a set of prefixes between HP(j) and HP(k) with j = 8i + 1,k=8i+7and i ∈[0,7].

In our paper, expanded prefixes are: EP(9, 15)、EP(17, 23)、EP(25, 31)、EP(33, 39)、EP(41, 47)、EP(49, 55) and EP(57, 63), and so on. The whole IPv6 route table is composed of the set of Hash prefixes and the set of expanded prefixes.

Definition 3-3: EPH(n) (expanded prefix Hash segment) and EPE(n) (expanded prefix expanded segment): a prefix with length n can be divided into two parts: high i bits and low (n-i) bits, where i is the maximum number that can be divided by 8 and smaller than n. The high i bits of expanded prefix is called EPH(n) and the low (n-i) bits is called EPE(n).

For example, a expand prefix 2a01:c */20, its EPH(20) is 2a01, EPE(20) is c.

Definition 3-4: HR(i) (Hash RAM). HR(i) is a set of RAMs called Hash RAMs store HP(i) and EPH(n), where i could be 16,24,32,40,48,56,64 and n is the length of the prefix.

Definition 3-5: ER(j,k)(expanded RAMs). ER(j,k) is a set of RAMs called expanded RAMs that store EPE(n) with n is between j and k, j = 8i + 1,k=8i+7and i ∈[0,7].

C. General structure of algorithm

In our proposed scheme, all HP and EPH are stored into HR called first level and EPE are stored into ER called second level, as shown in figure 1. Prefixes with different length will be stored into the first level or the second level. Hash conllisions will certainly occur when using Hash function, we use CAM to deal with them.

The algorithm we proposed includes four parts: first level lookup machine, second level lookup machine, priority comparator and CAM lookup machine (Hash confliction processor). In the following we will describe them in details.

First level lookup machine: In our paper, the first level lookup is Hash lookup. All HP and EPH are stored in this level.Every lookup process will go through this level.

Second level lookup machine: Second level lookup is expanded lookup. Only EPE are stored into this level in our paper.

Priority comparator: in the process of finding the best match, if a destination address finds more than one match, it needs to do comparison of priority.

For example, a destination address A finds three matching prefixes, the first is abc/ 12, its next hop is X, the second is abcd/16, its next hop is Y, the third is abcde/20, its next hop is Z. So after priority comparator, the last matching next hop is Z.

Handling with Hash conflictions: if Hash conflictions happen, the prefixes must be stored into CAM to ensure the accuracy of search when building route table, lookuping and updating the table,.

The key issues to be resolved in this paper are listed and explained in the following five parts.

D. Organization of Hash RAM and expanded RAM

The organization of HR and ER are illustrated as figure 2. The depth of HR(i) is 16, and the data width is 9+i. In the next, we will explain the Hash table and expanded table in details.

(1) The storage structure of HR(i)

The storage structure of HR(i) is illustrated in figure 2. Four types of data will be stored into HR, they are flag part, prefix part, next hop part and index part.

Flag part: two bit. The high bit (F) is a symbol of Hash conflictions, and low bit (E) represents whether the prefix has an expanded part.

Next hop part: next hop value of route table.

Index part: base address of the next level. The next level has seven RAMs, and the seven RAMs has same base address, which means that the expanded prefixes with same EPH have the same index value. For example, an EP(j,k) with length l, index value x and RPE value y, the address of next level ER is x*2l-j+1+y.

(2) Structure of expanded prefixes

The structure of expanded prefixes is illustrated in figure 2. It is consisted of flag part and next hop part.

Flag part: one bit. It is used to mark the item is empty or full.

Next hop part: the next hop of expanded prefixes.

E. XOR Hash function

We refer to the analysis [13] in order to determine which Hash function to choose. We choose XOR folding finally, which is simple to implement and has an excellent performance. The depth of Hash RAM is essential to Hash conflict rate. Figure 3 shows the relationship between Hash conflict and the depth of Hash RAM.

F. Building forwarding table

The length of HR(i) is determined by the number of prefixes, length of which is the same as i. As shown in figure 2, the first three part of Hash item is easily determined and the index part is determined by a counter we set in the program, which counts the number of EP with different EPH. Forwarding table is built using the algorithm shown in figure 4.

G. Algorithm of lookup

The process of constructing the routing table can be divided into three steps.

First step: Hash lookup. Searches in seven HRs are executed in parallel using the Hash indices obtained from XOR Hash function. As shown in figure 1, entries in each table are currently searched for the bucket indicated by Hash indices in the first level. Additionally, CAM is also searched in parallel. The result will be sent to priority comparator once it has a match.

Second step: expand lookup. Second level lookup is needed if expanded lookup is indicated in first level. Same as the first level, the 42 expanded RAMs are searched in parallel in second level, which uses the index part in the last level, as mentioned in previous. The matches will also be sent to comparator.

Third step: compare the received next hop value and output the longest matching next hop.

H. Update process

Routing table updated for the proposed scheme is the same as building process. The newly added prefix will be sent to CAM if confliction happens, otherwise sent to Hash RAM and expanded RAM in the light of the length of the prefix. Prefix deletion is similar to this. When the prefix needed to be deleted is in Hash table, we only need to delete the next hop part if "E" segment is high, else we need to delete the prefix part additionally. In other cases, delete all the contents of the certain location. If we delete an EP, the counter mentioned in previous is not change. By this way, the exist number can be reused by the newly added EP whose EPH part indicates the same HR location, and hence, fast and easy update is achieved.

IV. SIMULATION AND COMPARISON

This section presents the evaluation result of our proposed scheme. We implement the above data structure and algorithm with verilog HDL on a Xilinx Virtex-6 LVX240T chip and develop a verification platform. Both the design and verification are downloaded on a board FPGA ML605. The average lookup time is 1.07 memory access cycle. We have performed lookup simulation for our proposed scheme using data from Potaroo[12]. Table II shows the comparison between our proposed scheme and several existing scheme. [9] has lower storage efficiency and higher complexity compared with our scheme. Both [10] and [6] need special machinism to solve the storage problem. [11] achieve high-speed but is more expensive.

V. CONCLUSION

In this paper, we proposed a high performance two-level IPv6 lookup scheme that based on Hash, which can be used in national backbone network. However, the work we will do in the future is to reduce the Hash confliction rate. We can choose another Hash function or increase the depth of Hash table.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (60976020)

REFERENCES

[1] Ruiz-Sanchez,MA.;Biersack,EW.;Dabbous,W..Survey and taxonomy of IP address lookup algorithms[J].IEEE network,2001,PP:8-23.

[2] Li Y K.; Pao D.. Address lookup algorithms for IPv6[J]. IEE Proceedings-Communications, 2006, PP: 909-918.

[3] Yin,Keung Li;Pao, D..Comparative studies of address lookup algorithms for IPv6[J].Advanced Communication Technology, 2006,PP:285-290.

[4] Singh,M.; Garg, D..Choosing Best Hashing Strategies and Hash Functions. Advance Computing Conference,2006,PP:50-55.

[5] Gamache, B.;Pfeffer, Z.;Khatri, S P..A fast ternary CAM design for IP networking applications.Computer Communications and Networks, 2003,PP:434-439.

[6] Gupta P, Lin S, Mckeown N. Routing lookups in hardware at memory access speeds[C]. Citeseer, 1998.

[7] Erdem, O.;Bazlama, X E.;Ci, C F..Array Design for Trie-based IP Lookup.Communications Letters, IEEE,2010,PP:773-775.

[8] Zheng K.; Liu Z.; Liu B.. A scalable IPv6 route lookup scheme via dynamic variable-stride bitmap compression and path compression[J]. Computer Communications, 2006, PP: 3037-3050.

[9] Chu Y, Su H.; Lin P.; et al.. IC Design of IPv6 Routing Lookup for High Speed Networks[J]. Lecture Notes in Computer Science. 2006, PP:1832-1835.

[10] Li Z.;Deng X.; Ma H..Divide-and-conquer: a scheme for IPv6 address longest prefix matching[C].Workshop on High Performance Switching and Routing,2006,PP:37-42.

[11] Gamache B.; Pfeffer Z.; Khatri S P.. A fast ternary CAM design for IP networking applications[C]. Computer Communications and Networks, 2003,PP:434-439.

[12] Table Data. http://bgp.potaroo.net. 2011.

[13] Rai Jain.. A Comparison of Hashing Schemes for Address Lookup in Computer Networks[J]. IEEE TRANSACTIONS ON COMMUNICATIONS,1992.

TABLE I. THE NUMBER AND RATIO OF PREFIXES OF IPV6

Prefix	length(bits)	number	ratio (%)	trend
Can be divided by 8	8/16/24	1/1/1	0.06/0.06/0.06	slow
	32	1176	65.44	fast
	48	434	24.15	fast
	40/56/64	24/6/34	1.34/0.33/1.95	fast
Cannot be divided by 8	other	120	6.68	slow
Total number	1-64	1797	100%	fast

TABLE II. COMPARISON BETWEEN PROPOSED SCHEME AND EXISTING SCHEME

Lookup scheme	Entry numbers	Lookup cycle	Memory requirements
proposed	65535	1-2	RAM：15.33MB，CAM：28.75kB
[9]	150000	1-2	RAM：29.29MB，TCAM：20.04kB，CAM：10.24kB
[10]	60000	1-3	RAM：20-26MB
[6]	total	1-2	RAM：33MB
[11]	21632	15ns	TCAM：21Mb

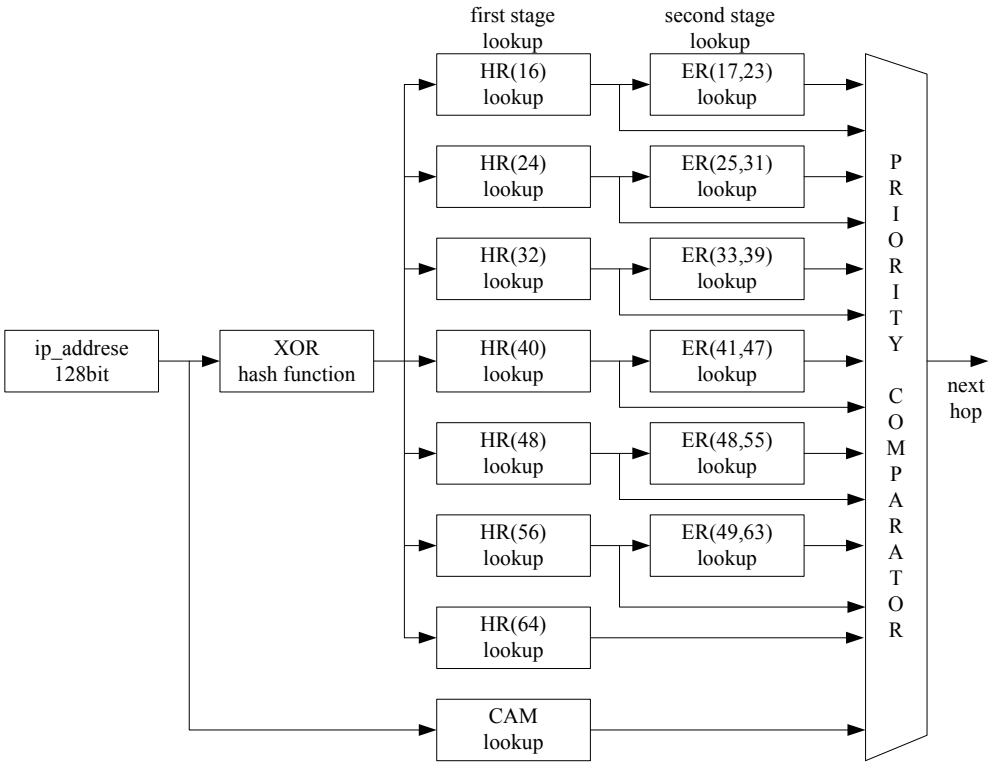


Figure 1 search procedure of forwarding table

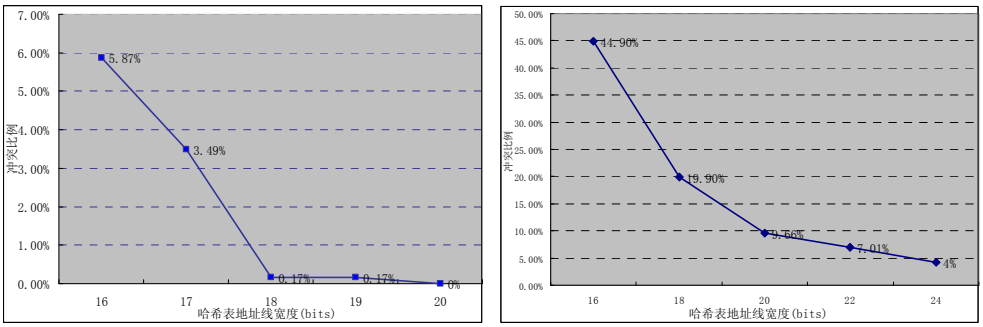
Storage structure of HR

FLAG	Prefix	Nh(i)	index

Storage structure of ER

H	Nh(i,i+6)

Figure 2 storage structures of Hash RAMs and expanded RAMs



(a) Hash conflict of 1800 entries (b) Hash conflict of 65536entries

Figure 3 relationships between Hash conflict and Hash depth

```
For prefix,length l P[l-1:0]
If (HP)
  XOR hash function
  Read hash item
  If(F=0)
    F=1,prefix part=prefix,Nh(i)= next hop
  If(F=1)
    Send the prefix to CAM
Else If (EP)
  EPH XOR hash function
  Read hash item
  If(E=0)
    E=1,index part=counter,counter added,
    Expanded ram pointer=counter
  If(F=1)
    Expanded ram pointer=index part
```

Figure 4 algorithm of building forwarding table