A Priority TCAM IP-Routing Lookup Scheme

Po-Chou Lin and Chung-Ju Chang, *Senior Member, IEEE*

***Abstract*—A priority TCAM IP-routing lookup scheme, which combines a priority ternary content addressable memory (TCAM) technique with a compact IP-routing lookup scheme, is proposed in this letter. It not only completes an IP-routing lookup with two memory accesses but also achieves small lookup table size and fast table reconstruction time.**

***Index Terms*—IP-routing lookup, ternary content addressable memory (TCAM).**

I. Introduction

HARDWARE IP-routing schemes can be classified into three categories: mutliway search, direct TCAM match, and indirect lookup. Lampson, Srinivasan, and Vargheseet [1] showed how multiway search can be adopted for solving the IP-routing problem. Its advantage is the small lookup table size. However, it suffers the disadvantage of large memory accesses per lookup and long routing table reconstruction time. Direct TCAM matching ASICs are available recently [2]. But TCAM with sufficient capacity for IP routing is still not cost effective. Gupta, Lin, and Mckeown [3] proposed an indirect lookup mechanism with 9 Mbytes memory and three memory accesses per lookup. Huang and Zhao [4] decreased the next hop array (NHA) size to 470 Kbytes. Wang, Chan, and Chen [5] further reduced the lookup time to two memory accesses with a little more memory.

In this letter, we propose a *priority TCAM IP-routing lookup scheme*, which combines a priority TCAM technique with a compact IP-routing lookup scheme. The priority TCAM IP-routing lookup scheme can complete an IP routing lookup with two memory accesses, reduce IP-routing lookup table size, and especially, decrease IP-routing lookup table reconstruction time.

II. The Priority TCAM IP-Routing Lookup Scheme

Although the number of Internet hosts grows exponentially, the routing prefixes with a router are still in sparse distribution. There are less than 45 000 routing prefixes over total 2^{16} segments in today's backbone routers. And except the default routing entry, there are few or even no routing prefix to define the next hop for most segments. Thus, we utilize this sparse distribution property to reduce IP-routing lookup table size.

Manuscript received December 20, 2002. This work was supported by the National Science Council, Taiwan, R.O.C., under Contract 91-2219-E-009-034, and by Lee and MTI Center for Networking Research at National Chiao Tung University, Taiwan, R.O.C., under Grant Q.528.

P.-C. Lin is with the Wireless Communication Technology Laboratory, Telecommunication Laboratories, Chunghwa Telecom Company Ltd., Taoyuan 326, Taiwan, R.O.C. (e-mail: pochou@cht.com.tw).

C.-J. Chang is with the Department of Communication Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: cjchang@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/LCOMM.2003.814709

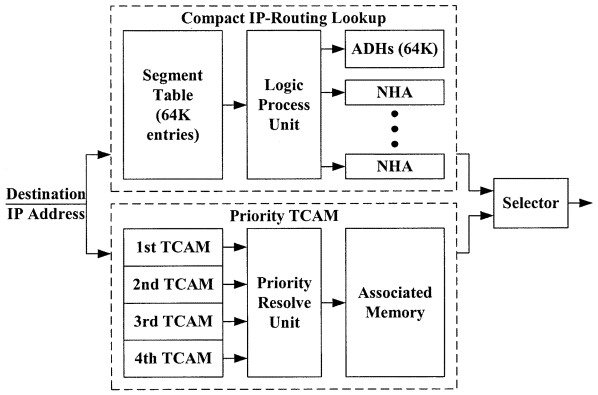

Fig. 1. Block diagram of the priority TCAM IP-routing lookup scheme.

TABLE I
Routing Prefixes of the 192.168 Segment

Entry i	Routing Entry $p_i/l_i/h_i$	$p_i(x,y)$ x=17,y=24	Compact IP-routing lookup or priority TCAM
0	192.168/16/0	xxxxxxxxb	Compact IP-routing lookup
1	192.168.20/22/1	000101xxb	Compact IP-routing lookup
2	192.168.84/22/2	010101xxb	Compact IP-routing lookup
3	192.168.68/23/3	0100010xb	Compact IP-routing lookup
4	192.168.68.16/28/4	01000100b	priority TCAM
5	192.168.68.16/32/5	01000100b	priority TCAM

Let l_i and h_i be the length and the next hop of a routing prefix p_i, respectively, and $p_i(x,y)$ represent the bit pattern from the xth bit to the yth bit in p_i. Table I shows an example of routing prefixes of the 192.168 segment, where entry 0 is the default prefix. Due to the maximum length $l_5 = 32$, the segment requires $2^{(32-16)}$ NHA entries if using Huang's algorithm. In this letter, the proposed *priority TCAM IP-routing lookup scheme* partitions the routing entries at prefix length equal to 24 and processes those routing entries greater than 24 by a TCAM technique, which will be discussed later. Fig. 1 shows the block diagram of the proposed priority TCAM IP-routing lookup scheme. The *compact IP-routing lookup* block is designed to process the IP-routing entries with $l_i \leq 24$. The *priority TCAM* block is designed to process the IP-routing entries $l_i > 24$. Therefore, at most $2^{(24-16)}$ NHA entries are required for the compact Ip-routing lookup. For entries entries 1 to 3 in Table I, the 24th bit of routing prefixes can be ignored, and NHA entries can be further reduced to $2^{(24-16-1)}$. Moreover, by further examining the pattern $p_i(17,24)$ (except the default prefix) with $l_i \leq 24$, the 17th, 19th, 21st, and 22nd bits are the same. This means that only a 3-bit (the 18th, 20th and 23rd bits) pattern with 2^3 NHA entries is needed to record. Notice that the more common bits there are, the fewer NHA entries will be. If both the compact IP-routing lookup block and the priority TCAM block match the incoming destination IP address, the *selector* block chooses the priority TCAM due to its longer prefix length.

A. Compact IP-Routing Lookup

Most of the routing entries are with length $16 < l_i \leq 24$. The compact IP-routing lookup block effectively compresses the memory required for these routing entries. It consists of the segment table, the logic process unit, the associated default hops (ADHs) and the NHAs, as shown in Fig. 1. The segment table contains 2^{16} entries and each entry records 4-byte segment information. The logic process unit analyses the incoming destination IP address, decides where the next hop is placed (ADHs or NHAs), and generates the corresponding address. The ADH is composed of 2^{16} entries, which stores the default hop of each segment and the NHAs contain the encoded next hop arrays.

The 4-byte segment information consists of 5 fields: 6 bits of common prefix pattern (Cprefix), 6 bits of common prefix position marker (Cmarker), 3 bits of maximum length of routing prefixes (Mlength), 1 bit of NHA encoding bits (N-bit), and 16 bits of NHA pointer (Npointer). Cprefix records the bits of the common bit pattern of $p_i(17,22)$; Cmarker sets the bits of the common bit position of $p_i(17,22)$ to 1 and resets to 0, otherwise; Mlength equals to the longest prefix length minus 17; Nbit indicates the number of NHA encoding bits in the NHA, where 0 and 1 represents the encoding of 4 and 8 bits, respectively; and Npointer points to the associated NHA memory. With double word alignment of memory space, Npointer can address up to 256 kbytes NHA memory space; it cannot be set to zero except that only default route exists on that segment. Under such circumstances, ADH records the default hop of that segment and no corresponding NHA is needed. Take entries number from 0 to 3 in Table I as an example. The 17th, 19th, 21st, and 22nd bits are of the same, which results in $\mathrm{Cmarker} = 101\,011\mathrm{b}$ and $\mathrm{Cprefix} = 0\mathrm{x}0\mathrm{x}01\mathrm{b}$, where x represents don't care. The longest prefix length classified to the compact IP-routing lookup is 23, so Mlength is equal to 6. We use 4-bit coding to encode the output ports ranging from 0 to 3, so Nbit is set to 0. The NHA port map is {0,0,1,1,0,3,2,2} and requires only 4 bytes memory space.

For more clear explanation, *Virtual Codes 1* and *2* are shown in the following to describe the compact IP-routing lookup table construction algorithm and its operation algorithm, respectively. Here let $V(p_i(x,y))$ represent the value of bit pattern $p_i(x,y)$ and Clength counts the total number of 1's in Cmarker. Define two operators $\mathbf{P0}(\cdot)$ and $\mathbf{P1}(\cdot)$ on $X_0X_1\ldots X_s$ and $r_0r_1\ldots r_s$ by: $\mathbf{P0}(\mathbf{X_0}\ldots\mathbf{X_s},\mathbf{r_0}\ldots\mathbf{r_s}) = X_{\sigma_0}\cdots X_{\sigma_t}$, where $\sigma_0 < \cdots < \sigma_t \leq s$, for all $r_{\sigma_i} = 0$; and define $\mathbf{P1}(\mathbf{X_0}\ldots\mathbf{X_s},\mathbf{r_0}\ldots\mathbf{r_s}) = X_{\sigma_0}\ldots X_{\sigma_t}$, where $\sigma_0 < \cdots < \sigma_t \leq s$, for all $r_{\sigma_i} = 1$. For example, $\mathbf{P0}(X_0\ldots X_7, 10\,101\,100\mathrm{b}) = X_1X_3X_6X_7$ and $\mathbf{P1}(X_0\ldots X_7, 10\,101\,100\mathrm{b}) = X_0X_2X_4X_5$. Also, "$\ll$" ("$\gg$") means left (right) shift operation. From *Virtual Codes 1* and *2*, the table reconstruction complexity is in $O(m)$ and the IP-routing lookup can be accomplished with two memory accesses.

Virtual Codes 1: Lookup Table Construction Algorithm

Input: The set of routing prefixes in a segment

Output: The corresponding segment entry, ADH and NHA of this segment

Step 1: Let $\mathrm{P} = p_0, p_1, \ldots, p_{m-1}$ be the set of sorted prefixes of an input segment; For any pair of prefixes p_i and p_j in the set, $i < j$ if and only if $l_i < l_j$; p_0 is the default routing prefix with $l_0 = 16$ and default hop h_0;

Step 2: Set the ADH entry to h_0;
If $(m = 1)$ {Npointer $= 0$; goto Step 7;}

Step 3: Assign $\mathrm{Mlength} = l_{m-1} - 17$; $\mathrm{Cprefix} = p_1(17,22)$;
$\mathrm{Cmarker} = 111\,111\mathrm{b}$; $\mathrm{Nbit} = 0$;

Step 4: For $j = 1$ to $m-1$ do
$\mathrm{Cmarker} = (.\mathrm{not.}\ (p_j(17,22)\ .\mathrm{xor.}\ \mathrm{Cprefix}))$ .and. Cmarker;
if $(h_j > 15)$ $\mathrm{Nbit} = 1$;
End

Step 5: Get $2^{(Mlength-Clength)}/2^{(1-Nbit)}$ bytes free memory space with starting address Addr;
Fill the NHA space with default hop h_0;
$\mathrm{Npointer} = \mathrm{Addr} \gg 2$;

Step 6: For $\mathrm{j} = 1$ to $\mathrm{m} - 1$ do
$\mathbf{k} = (\mathbf{P0}(p_i(17,24), (\mathrm{Cmarker} \ll 2)) \gg (7 - \mathrm{Mlength}))$;
From the kth to the $(k+2^{(Mlength-(l_j-17))}-1)$th entries are updated with h_j;
End

Step 7: Stop ■

Virtual Codes 2: Lookup Table Operation Algorithm

Input: The incoming destination IP address p_i

Output: The next hop of compact IP-routing lookup

Step 1: $\mathrm{j} = p_i(1,16)$;
Get Cprefix, Cmarker, Mlength, Nbit and Npointer from the jth entry of the segment table;

Step 2: $\mathrm{P} = \mathbf{P1}(\mathrm{Cprefix},\ \mathrm{Cmarker})$;
$\mathrm{P1} = \mathbf{P1}(p_i(17,22), \mathrm{Cmarker})$;
$\mathrm{k} = (\mathbf{P0}(p_i(17,24), \mathrm{Cmarker} \ll 2) \gg (7 - \mathrm{Mlength}))$;

Step 3: If $((\mathrm{Npointer} \neq 0)\ .\mathrm{and.}\ (\mathbf{P1} = \mathbf{P}))$
next hop = content of the jth entry from address (Npointer $\ll 2$);
Else
next hop = content of the jth entry of ADHs;
End

Step 4: Stop ■

B. Priority TCAM

The priority TCAM block, shown in Fig. 1, consists of TCAMs, the priority resolve unit, and the associated memory. Each TCAM generates matched address for the associated memory. The TCAM, as shown in Fig. 2, is composed of the prefix p_i register, the corresponding mask bit register induced from the prefix length l_i, 32 3-input comparators, and a 32-bit AND operation. The i-th TCAM has higher priority than the j-th TCAM, if $i < j$. The priority resolve unit selects the highest priority matched output from the TCAMs. And associated memory stores the next hop h_i of the associated TCAM entry.

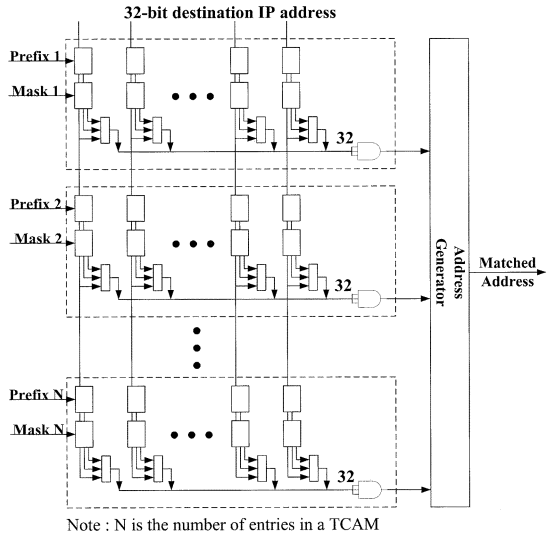

Fig. 2. The architecture of each TCAM.

The routing entries stored in each TCAM must be carefully arranged so that at most one entry of each TCAM can be matched for any incoming destination IP address. Due to the nonevenly distribution of l_i, we implement 4 priority classes, instead of 8, with each class corresponding to a specific prefix length for easy management and full utilization of the TCAM memory resource. There are rare cases that an incoming IP address matches more than four of TCAM entries, which cannot be placed into the four priority classes. The *routing entry prefix expansion technique* can solve this problem. For example, 8 routing entries ($192.168.68.10/25/0$, $192.168.68.10/26/1$, $\ldots$, $192.168.68.10/31/6$, $192.168.68.10/32/7$) cannot be fitted into 4 priority TCAMs. By the prefix expansion technique, the routing entry 192.168.68.10/ 25/ 0 can be expanded to 192.168.68.64/ 26/ 0 and 192.168.68.10/ 26/ 0. Merged with the routing entry 192.168.68.10/ 26/ 1, the expanded entries can be combined to a routing pair of (192.168.68.64/ 26/ 0, 192.168.68.10/ 26/ 1). By the same way, all the 8 routing entries can be expanded and merged into 4 pair entries of the same prefix length and fitted into 4 priority TCAM classes, respectively. Arranging IP addresses into different priority classes and using prefix expansion techniques are rarely happened and required only at the IP-table construction phase.

The priority TCAM lookup can be completed with one memory access plus few logic operation cycles, which is faster than the compact IP-routing lookup. An efficient TCAM architecture can be thus designed by releasing the TCAM lookup time from 1 clock cycle to more, say 4, clock cycles so that the lookup can be finished with two memory accesses. That is, about 75% gate count can be saved without sacrificing overall performance by replacing one operation of a 32-bit address comparator with four operations of a 8-bit IP address comparator.

III. Results and Discussion

For the routing scenario given in Table I, the Huang's algorithm [4] requires 1 to 3 memory accesses and about 8K lookup memory; the Chen's algorithm [5] requires 1 or 2 memory accesses and also about 8K lookup memory; and the proposed priority TCAM IP-Routing lookup scheme requires 2 memory accesses and 9 bytes lookup memory plus 2 priority TCAM entries. Table II shows the memory requirements of the Huang's algorithm, the Chen's algorithm, and our scheme under realistic prefix data of several network access points (NAPs) from the Internet performance measurement and analysis (IPMA) project [6]. It can be seen that the IP-routing lookup memory is significantly reduced.

TABLE II
Comparisons of Memory Requirements

	AADS	Mae-West	PAIX
Prefixes	33931	37523	18569
Segments	5813	6126	3571
Length>24	431	433	443
Huang's[4]	599K	610K	507K
Chen's[5]	659K	781K	543K
Priority TCAM IP-Routing Lookup	423K +431 TCAM	463K +433 TCAM	377K +443 TCAM

Moreover, the table reconstruction time should be fast enough to handle the frequent route updates (e.g., 100 route updates per second) in a backbone router. The algorithms [3], [4] suffer from long table reconstruction time because they may require $2^{(32-16)}$ modifications of table content for one route update in the worse case. The proposed priority TCAM IP-routing lookup scheme can complete a route update within $2^{(24-16)}$ modifications if $l_i \leq 24$, or just by adding a new TCAM entry if $l_i > 24$. For memory access cycle of 10 ns, it can complete 100 routing updates within 64 μs ($100 \cdot 2^{(24-16)} \cdot 10$ ns) in the worse case, such that the switch performance of the IP router can be maintained. As a matter of fact, the faster routing table update can be accomplished by storing all the updates in the priority TCAM for the time being. In other words, the router can put all the recently exchanged IP-routing prefixes in the priority TCAM block temporarily, and reconstructs the corresponding compact IP-routing lookup table later when the router is available for updating or the priority TCAM memory is congested. Consequently, The priority TCAM IP-routing lookup scheme can significantly reduce IP lookup memory size, minimize IP lookup table reconstruction time, and complete an IP lookup with two memory accesses.

References

[1] B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," *IEEE/ACM Trans. Networking*, vol. 7, pp. 324–334, June 1999.
[2] 9M TCAM SCT9000 by SiberCore Technologies [Online]. Available: http://www.sibercore.com/products. htm
[3] P. Gupta, S. Lin, and N. Mckeown, "Routing lookups in hardware at memory access speeds," *Proc. IEEE INFOCOM'98*, Mar. 1998.
[4] N. F. Huang and S. M. Zhao, "A novel IP-routing lookup scheme and hardware architecture for multi-gigabit switching routers," *IEEE J. Select Areas Commun.*, vol. 17, pp. 1093–1104, June 1999.
[5] P. C. Wang, C. T. Chan, and Y. C. Chen, "A fast IP routing lookup scheme," *IEEE Commun. Letter*, vol. 5, pp. 125–127, Mar. 2001.
[6] Internet Performance and Analysis (IPMA) Statistics and Daily Reports. Merit networks, Inc. [Online]. Available: http://www.merit.edu/ipma/routingtable/