# A Fast IP Routing Lookup Scheme

Pi-Chung Wang, Chia-Tai Chan, and Yaw-Chung Chen, *Member, IEEE*

***Abstract*—A major issue in router design for the next generation Internet is the fast IP address lookup mechanism. The existing scheme by Huang *et al.* performs the IP address lookup in hardware in which the forwarding table can be compressed to fit into reasonable-size SRAM, and a lookup can be accomplished in three memory accesses. In this letter, we claim that with a little extra memory, it is able to further reduce the lookup time to two memory accesses.**

***Index Terms*—Gigabit networking, Internet, IP address, lookup.**

## I. Introduction

IP ROUTES have been identified by a ⟨routing prefix, prefix length⟩ pair [5], where the prefix length ranges from 1 to 32 bits. The most straightforward way to perform a lookup is to build a forwarding table in hardware for all possible IP addresses. However, the size of the forwarding table (next-hop array; NHA) would be too huge ($2^{32}$ entries) to be practical. An indirect lookup mechanism with 9-Mbytes memory and three memory accesses per lookup has been proposed in [2]. In [3], Huang *et al.* reduced the NHA size to 470 kbytes based on the distribution of the prefixes within a segment.

In this letter, we further reduce the IP address-lookup time to two memory accesses. Although the required memory space slightly increases, it can be justified by the nowadays SRAM cost. The rest of this letter is organized as follows. Section II presents the proposed scheme and the related issue of pipelining hardware. The performance analysis is addressed in Section III. Section IV concludes the work.

## II. The Proposed Scheme

Although the number of Internet hosts grows exponentially, the routing prefixes within a router are still in sparse distribution. In fact, there are approximately 45 000 routing prefixes out of a total of 65 536 segments in today's backbone routers. We observed that only few segments contain multiple routing prefixes. For most segments, there are fewer or even no routing prefix to define the route. Thus, we can properly arrange prefixes for these segments to reduce the required memory. From the sample prefixes in Fig. 1, it requires $2^{(24-16)} = 2^8$ NHA entries using Huang's algorithm. By examining these prefixes, we can find that their first 17 bits ⟨0 001 100 000 110 000⟩ are the same. This means we only have to record 7-bit (the 18th bit to the 24th bit) patterns of these prefixes with $2^{(24-16)} = 2^7$-entry NHA. Obviously, the more common bits there are, the fewer NHA entries will be.

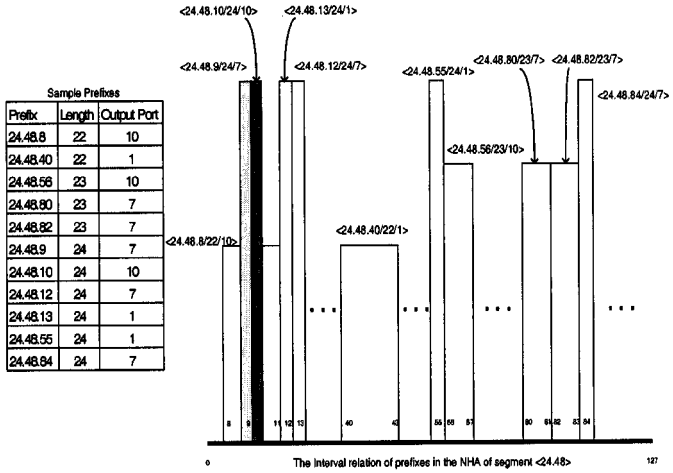


Fig. 1. NHA construction example.

By tracing the Mae-Eat NAP router [4], we found that approximately 54% of the segments use less than two output ports, and 96% of the segments use less than four output ports. Thus we could use fewer bits to encode the output port identifier in an NHA. At first, we use a bit-vector to collect all possible output port identifiers, which are then encoded using the smallest port identifier as the $base$. Each associate NHA entry will be set to the value of its output port identifier minus the *base.* Two extra fields $cbits$ and $base$ are appended to the entry of the segment table. Thus the $base$ can be read before indexing the associated NHA, and the physical output port index can be calculated by adding the $base$ index to the NHA indexing result. This process can be completed within two memory accesses plus the calculation time which can be ignored.

The NHA construction algorithm for a segment is given below. Let $l_i$ and $h_i$ be the length and the output port identifier of a routing prefix $p_i$, respectively. The $cprefix$ represents the common bits of routing prefixes beyond the first 16 bits in a segment, and $clength$ is the valid length of $cprefix$. $Mlength$ is equal to the longest prefix length minus 16, thus $clength \leq Mlength$. Let $p_i(x, y)$ represent the bit pattern from the $x$th bit to the $y$th bit in $p_i$. The entries from $V(p_i(clength + 26, l_i)) \times 2^{Mlength-(l_i-16)} - 1)$ to $V(p_i(clength + 26, l_i)) \times 2^{Mlength-(l_i-16)} + (2^{Mlength-(l_i-16)} - 1)$ will be updated with $h_i$ minus *base*, where $V(p_i(clength + 26, l_i))$ represents the value of bit pattern $p_i(clength + 16, l_i)$. Taking the prefix ⟨24.48.40/22/1⟩ in Fig. 1 as an example, obviously $Mlength$ is 8 and $l_i$ is 22. So we can calculate that $V(p_i(clength+26, 22)) \times 2^{8-(22-16))} = 40$, where $clength = 1$, and the number of updated entries is $2^{8-(22-16)} = 4$. There are four entries (the 40th entry to the

Manuscript received February 17, 2000. The associate editor coordinating the review of this letter and approving it for publication was Prof. N. Shroff.

P.-C. Wang and Y.-C. Chen are with the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. (e-mail: pcwang@csie.nctu.edu.tw; ycchen@csie.nctu.edu.tw).

C.-T. Chan is with the Telecommunication Laboratories, Chunghwa Telecom Company, Ltd., Taipei, Taiwan, R.O.C. (e-mail: ctchan@ms.chttl.com.tw).

Publisher Item Identifier S 1089-7798(01)03558-X.

43rd entry) in the NHA to be updated. Since most routers have a default route with zero prefix length which matches all addresses, the table entries would be assigned the value of the default route initially.

### A. NHA-Construction Algorithm

**Input:** The set of routing prefixes of a segment.
**Output:** The corresponding NHA of this segment.
*Step 1)* Let $P = \{p_0, p_1, \ldots, p_{m-1}\}$ 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 $l_i < l_j$.
*Step 2)* Assign $cprefix = p_0(17, i_0)$, $clength = l_0 - 16$ and $Mlength = l_{m-1} - 16$.
*Step 3)* For $i = 0$ to $m - 1$ do
  $cprefix$ = common bits between $cprefix$ and $p_i(17, l_i)$.
  $clength$ = valid length of $cprefix$.
*Step 4)* Calculate the $cbits$ and $base$ from the vector.
*Step 5)* Construct the NHA with size $2^{Mlength-clength} \times cbits$ and initialize it with default route.
*Step 6)* For $i = 0$ to $m - 1$ do
  calculate the range of updated entries and fill them with the output port index minus $base$.
*Step 7)* **Stop.**

The time complexity of the above algorithm is $O(m \log m)$, which is bounded by the prefix sorting. Upon receiving a new routing prefix, Step 3 will be repeated and NHA will be rebuilt if necessary. By applying the proposed algorithm, the number of entries in the generated NHA can be 25% less than that in Huang's algorithm. Although the table compression ratio in our proposed scheme is not so notable, it is able to accomplish an IP lookup with two memory accesses. Also the proposed scheme is implementation feasible with pipelining hardware. Notice that in Huang's algorithm, the memory locations accessed in the second and the third lookup are the same because it appends the CNHA to the tail of CWA [3]. Therefore, it may cause the structural hazard for implementation in pipelining hardware. Such potential structural hazard can be avoided in our proposed scheme without requiring any specific hardware such as dual port memory.

The entry of the segment table consists of 6 fields: pointer/next hop, $cprefix$, $clength$, $Mlength$, $base$, and $cbits$. The length of pointer/next hop is 20 bits which can map up to 1 Mega memory addresses. Since the maximum prefix length minus the length of the segment (16) is smaller than 16, the number of bits needed for $Mlength$ is 4. Trivially, the $base$ is 8 bits and the maximum number of encoded bits is 8. Thus the bit count of $cbits$ is 3. If we use $cprefix$ with length 3, then the $clength$ is 2 bits. As a consequence, if an NHA entry is 40 bits, it will result in the segment table size $2^{16} \times 40$ bits, i.e., 320 kbytes.

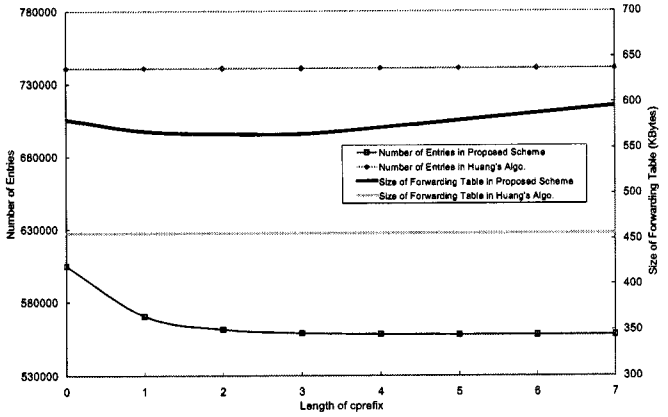


Fig. 2. Effect of the $cprefix$ length.

TABLE I
THE COMPARISON OF MEMORY REQUIREMENTS

| Site | Routing Prefixes | Proposed Scheme (Kbytes) | Hunag's Algo (Kbytes) |
|---|---|---|---|
| AADA | 35,769 | 422 | 425 |
| Mae-East | 44,075 | 565 | 456 |
| PacBell | 24,773 | 320 | 396 |
| Paix | 9,335 | 364 | 331 |
| Mae-West | 29,973 | 530 | 424 |

## III. PERFORMANCE ANALYSIS

Through simulation, we show that the proposed scheme features low memory requirement while achieving very high IP lookup performance. We use the real world data obtained from the IPMA project [4] as a basis for comparison, these data provide a daily snapshot of the routing tables used by some major Network Access Points (NAP's).

In Fig. 2, we demonstrate the effect of the $cprefix$ through the trace available in the router of the Mae-East NAP. By coupling the effect of $cprefix$, the resulted number of entries is 25% less than that in Huang's scheme. Also the size of the forwarding table in our scheme is no larger than 565 kbytes. While the length of $cprefix$ increases, the number of NHA entries decreases.

In Table I, we log five traces to build the forwarding table. Although the size of the forwarding table might become larger as a trade-off for throughput enhancement, we can find that the required memory size does not increase too much, or even decrease in two traces (AADS and Pac-Bell). This is because that the use of $cprefix$ reduces the number of NHA entries effectively. While in the traces of two backbone NAP's (Mae-East and Mae-West), the memory increment is notable, this is due to the diverse routing prefixes in the backbone, in which the effect of $cprefix$ is degraded.

## IV. CONCLUSIONS

In this letter, we propose a fast IP-address lookup scheme which is implementation feasible with nowadays high-speed SRAM. Our scheme can complete a lookup with two memory accesses. From the simulation results, the required memory is no larger than 565 kbytes. With two memory accesses for an IP

lookup, the proposed scheme can avoid the structural hazard in hardware pipelining. By applying state-of-the-art SRAM (i.e., 5 ns access time) technology, our scheme is able to achieve more than a hundred million routing lookups per second.

## REFERENCES

[1] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, "Small forwarding tables for fast routing lookups," in *Proc. ACM SIGCOMM'97*, Cannes, France, Sept. 1997, pp. 3–14.

[2] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proc. IEEE INFOCOM'98*, San Francisco, USA, March 1998.

[3] N. Huang, S. Zhao, and J. Pan, "A fast IP routing lookup scheme for gigabit switch routers," in *Proc. IEEE INFOCOMM'99*, New York, NY, March 1999.

[4] Merit Networks, Inc., "Internet performance measurement and analysis (IPMA) statistics and daily reports,", http://www.merit.edu/ipma/routingtable/.

[5] Y. Rekhter and T. Li, "An architecture for IP address allocation with CIDR,", RFC 1518, Sept. 1993.

[6] M. Waldvogel, G. Vargnese, J. Turner, and B. Plattner, "Scalable high speed IP routing lookups," in *Proc. ACM SIGCOMM'97*, Cannes, France, Sept. 1997, pp. 25–36.