IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 8, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

IP address lookup for Internet routers using cache routing table

Houassi Hichem[1] **and Bilami Azeddine**[2]

[1]**Department of Computer Science, University Center of Khenchela, Algeria**

[2] **Department of Computer Science, University of Batna, Algeria**

Abstract

So that the routers forward an IP packet with his destination, they are running a forwarding decision on an incoming packet to determine the packet's next-hop router. This is achieved by looking up the longest matching prefix with a packet destination address in the routing tables. Therefore, one major factor in the overall performance of a router is the speed of the IP address lookup operation due to the increase in routing table size, in this paper, a new IP address lookup algorithm based on cache routing-table is proposed, that contains recently used IP addresses and their forwarding information to speed up the IP address lookups operation in the routers. We evaluated the performance of our proposed algorithm in terms of consultation time for several sets of IP addresses, the results of performance evaluation show that our algorithm is efficient in terms of the lookup speed since search can be immediately finished when the input IP address is found in the cache routing table.

Keywords: *Internet router, cache routing table, IP addresses lookup, longest prefix.*

1. Introduction

The Internet is becoming omnipresent and has been exponentially growing in size. The number of users, sub-networks and domains connected to the Internet seem to be exploding. The Internet network is comprised of routers, that forward packets towards their destinations, and physical links that transport packets from one router to another. Each router is required to perform a forwarding decision on an incoming packet to determine the packet's next-hop. This is achieved by looking up the destination IP address of the incoming packet in a forwarding table. The packet forwarding process in a router is to find the longest prefix in the routing table that provides the best match to the destination IP address of the data packet. When a router receives an IP packet, it searches for the longest prefix matches the destination IP address of this packet.

The routing table is structured in two main entry (prefix and prefix length), where the length of a prefix is limited by the length of the destination IP address (32 for the IPv4 addresses and 128 for IPv6).

For years, the main bottleneck in the performance of IP routers is the time to find a next-hop for an incoming IP packet in the routing table. The problem is defined as the research of the longest prefix from all prefixes that match the destination IP address in the routing table.

The rest of the paper is organized as follows. Section 2 describes the drawbacks to the existing approaches to IP lookups. Section 3 briefly defines the IP addresses lookup problem. Section 4 describes our lookup schemes based on cache routing table. Section 5 details the routing table structure and the cache routing table management. Subsection 5.1 represents the main routing table structure; Subsection 5.2 represents the cache routing table structure; Subsection 5.3 describes the entry replacement in the proposed cache routing table; Subsection 5.4 describes the cache routing table scheduling mechanism. Section 6 describes performance measurements of our schemes. We conclude in Section 7 by assessing the contributions of this paper.

2. Existing approaches to IP Lookup

Different hardware and software approaches are proposed and deployed to improve the performance of routers [8], the main metric of performance was the forwarding speed or the number of packets forwarded per second.

There are many approaches for fast IP-address lookup that are based on software solutions. The search for next hop in the routing table is in general, by the longest prefix technique, the binary trie [19] is the basis of the majority of these approaches. Several other variants of binary trie have been proposed to improve the longest prefix match search performance [1,3,7], The longest prefix search operation on an input destination IP address starts from the root node and recursively go to the left child or the right child until there is no more child to proceed. At each node, if the IP address matches the stored prefix, it is remembered as the current best matching prefix; one extension to the binary trie is the multibit trie [1] which reduce the depth of the binary trie, the research process in the

multi-bit trie skip several bits of the IP address at the same traversal between nodes. Authors of [2] proposes the level and path compression trie (LC-trie), the path compression removes internal nodes with only one child. With these methods, The LC-trie significantly reduces the depth and the number of nodes of the trie. Authors of [20] proposed a Binary prefix tree (BPT) algorithm attempts to perform binary search on prefix values. (BPT) is constructed by defining the comparison on values of two different length prefixes. This technique an excellent property in memory accesses due to the absence of internal empty nodes, but the depth of the tree depends on the number of prefixes. Since the empty nodes in the binary trie are not associated with any prefix then to remove empty nodes, the priority trie (P-trie) [21] relocates the longest prefix of the sub-trie rooted by an empty node into the empty node. Recently, binary search tree with prefix vector is proposed. This tree is constructed only with leaves of the binary trie and makes each leaf hold the nested prefix information using a prefix vector. This tree requires a wide memory because of the prefix vector. These data structures produce high-speed lookups for the 32-bit addresses of the IPv4 addresses but their performance degrades when they are used for the 128-bit addresses of the IPv6. Therefore this performance degradation is due to the increase in number of memory accesses as a result of the growth of the routing table;

Hardware based address lookup algorithms are classified into four classes, namely, the memory on a single application specific integrated circuit (ASIC) [5], the content addressable memories (CAMs) and ternary content addressable memories (TCAM) [6], reconfigurable hardware-based solutions and parallel architectures solutions [12] [13].

The ASICs are typically used to implement binary tree data structures using the high-speed memories such as SRAMs. A number of researchers have stated that the speed of the IP packets forwarding using these memories can be significantly increased [11]. To reduce the time for memory accesses, many hardware solutions use a pipelined architecture that allows one address lookup per memory cycle [10].

Trie-based IP lookup schemes usually require several memory accesses per lookup and those accesses may be serialized. In contrast, CAMs and TCAMs [6] can perform a lookup operation in a single cycle owing to its parallel access characteristics. Traditionally, TCAM and CAMs have been designed as a hardware solution to store prefixes to make consultations with IP addresses and IP packets classification. TCAM are fully associative memories in which each entry is appropriate to store fields of the prefixes, TCAM is designed so that all entries are compared in parallel, and then a TCAM is a memory used for searching a data item (bits of a prefix) in parallel.

Parallel architectures proposed in [12] [13] are based on the principle of partitioning the routing table into several sub-tables handled in parallel to find the longest prefix in each sub-table, and then select the most longest prefix between results in different sub-tables. The essential difference between these proposals architectures is the partitioning criterion of the routing table and the main principles of the longest prefix selection among the partial results obtained.

Other approaches are proposed to improve the performance of routers by using a cache memory to store recently referenced IP addresses and their forwarding information. All the future packets that would match the same prefix in the routing table will find their route in the prefix cache and will not need to refer to the routing table. The majority of these approaches as [4] [14] [15] [16] [17] [18] proposes hardware solutions i.e processors with memory that contains prefixes recently referenced. [14] Proposes an expansion of prefixes in the cache and search the IP address by comparison, [4] proposes a cache-prefixes architecture without expansion. [16] Proposes multizone pipelined cache architecture.

3. IP Address lookup problem

The IP (Internet Protocol) defines a mechanism for transmitting Internet packets where each packet has a destination IP address. In an Internet router, there is a routing table that associates any IP address to an output port (next hop) by the prefix technique. Because each entry in a routing table (prefix) may specify a sub-network, one destination address may match more than one routing table entry. The IP lookup operation becomes more and more computationally intensive because the variable sized prefixes have been extensively employed since the advent of CIDR(Classless Inter-Domain Routing) [9]. Since several prefixes can be matched for a destination IP address under the CIDR, the router has the burden to select the longest matching prefix (LMP) as the best matching one. Therefore, the performance of the router strongly depends on the efficiency of the IP lookup operation.

For example, the prefix 1101* matches all destination addresses that begin with 1101, and 10010* matches all destination addresses that begin with 10010. Suppose that a router table includes the prefixes P1 = 101*, P2 = 10010*, P3 = 01*, P4 = 1*, and P5 = 1010*. The destination address d = 1010100 is matched by the prefixes P1, P4, and P5. Since, length(P1) = 3 (the length of a prefix is a number of bits in the prefix), length(P4) = 1, and length(P5) = 4, P5 is the longest prefix that matches d. In longest-prefix routing, the next hop for a packet destined for the destination address d is given by the longest prefix that matches d.

The IP Address Lookup problem is to study how to construct a data structure for storing the routing table prefixes, so that we can quickly find the output port (next hop) for a data packets received to a router.

4. Router with a cache routing table

The router is composed of two routing tables in our proposition, a main routing table and a cache routing table. The cache routing table is relatively small and quick table, each entry in the cache routing table is composed of three fields, the IP destination address, the packet forwarding information such as the output port number of the router and the prefix of the IP address generated from the main table. The role of this cache table is to store the IP addresses recently used in the main routing table that is larger and slower. It serves to accelerate the road search operation, in order to find the output ports of the router quickly to send a data packet received by a router.

The diagram in Figure 1 illustrates the routing table of a router with a cache routing table. The routing table is composed of two tables, the main routing table and the cache routing table. When a packet arrives at a input port of a router, a router port extract the destination IP address and searches for the output port in the cache routing table, if the destination IP address extracted found, the data packet is forwarded to the output port of the router matching this IP address in the cache routing table, if the IP address does not exist in the cache routing table a new search is started in the main routing table using the search by prefix technique (longest prefix) [7].

Once the output port corresponding to the longest prefix of the IP address found in the main routing table, two main operations are performed:

(1) Forwarding of the data packet to the output port,

(2) Update the cache routing table by this new information; insertion of a new entry containing the IP address, the prefix found in the main routing table and the output port for future research in the cache routing table.

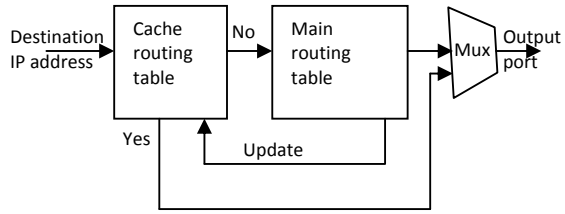

Fig. 1 Diagram of a router with a cache routing table

5. Routing table structure

5.1. Main routing table

In order to allow a more efficient use of the IP address space and avoid the problem of routing table explosion, prefixes in the main routing table can have an arbitrary length from 0 to 32 bits for IPv4 and from 0 to 128 bits for IPv6 using CIDR. To further reduce the routing tables, Table 1 shows an example of a main routing table, a table entry that contains the prefix, the prefix length and the output port. The Next hop research problem in the main routing table is to find a longest prefix matching for a destination IP address among all prefixes of the routing table.

Table 1: Main routing table

Prefix	*Prefix length*	*Next Hop*
11*	2	A
10*	2	B
111*	3	C
1101*	4	D

5.2. Cache routing table

Our cache routing table is used as the main routing table for fast next hop search.

During the IP packets forwarding operation, the router will search the routing table entry that matches the destination IP address of the incoming IP packet, to speed up the searching in the cache table, we proposed a table of limited size and contrary to the main routing table the research done by direct comparison of cache table entries with the destination IP address, for that, the entries of the proposed cache routing table are as follows:

- The complete IP address: to avoid the use of prefixes in the cache table.
- The prefix found in the main routing table. This information is used for the routing table updates operations
- The Next hop: The router interface to which the packets routed are sent.

Table 2: Cache table with IP addresses of 5 bits

IP Address	*Prefix*	*Next Hop*
11011	110*	A
10110	101*	B
11010	11*	C
11111	111*	D

5.3. Entry replacement in the cache routing table

Our approach includes not only the search operation of next hop, but also the operations necessary to update the cache routing table.

The cache table may not contain all the IP addresses searches in the main table because of its limited size, we must define a method indicating which cache table entry should be replaced by the new IP address. This method is called entry (IP address) replacement method of the cache table.

An IP address is restored in cache table when it is required and found in the main routing table. But the

problem of the withdrawal remains; what address will leave its place to the restored IP address?
The ideal is to remove an IP address just after its last use (corresponding to the last data packet routed) but this requires knowing the last data packet in a data flow.
To approach to the optimal algorithm must be based on the following observation: Since all packets in a same data flow arrived in a router has the same destination IP address and are transferred in sequence, once a user starts a download operation, the routers try to forward all data packets constituting the data flow continuously to their destination. This suggests the following mechanism: when a default address occurs in the cache table, the address that has not been used for the longest time is removed and replaced by the new address.
This method is called the least recently used. To fully implement, we propose to manage a linked list of all entries (IP addresses) of the cache table, with the most recently used address in top and the least recently used in queue of list, this list must be updated at each reference to the cache table

5.4. Cache table scheduling

At the arrival of a new data packet to the router, the router extracts the IP destination address and uses the IP destination address to look up the cache table to get the output port for this packet, and subsequently move this cache table entry to the header list of IP addresses. By this technique of moving entries, the most recently used IP address is located on top of the list and the least recently used is at the bottom of the IP addresses list.

Scheduling procedure:

```
Procedure cache_list_sheduling
{
  For all incomin data packet
{
      Adr ← packet IP address
       While (the IP addresses list IP_Adr_List
              Unfinished)
       {
         If (the address Adr found in the list
             List_Adr_IP) Then
         {
            Return the output port
            Move this item to the List_Adr_IP
            header
         }
       }
    }
}
```

6. Experimentation

Routing lookup algorithm can be evaluated by its lookup speed. To evaluate the effectiveness and performance of our proposed algorithm, we developed a generator which allows: To generate a routing table containing prefixes with various lengths, and allows us to vary the entries number of these tables, and on the other hand to generate a series of IP packets used like traces of IP routers
We used a database representing the routing table for our experiments, the objective of the first test is to vary the number of entries in the cache routing table to specify cache size optimizing the search time of the output port, using a set of 496965 IP packets and a main routing table with 10000 prefixes.
Concerning the evaluation of the next hop search operation, the routing table has received neither suppression non insertion of prefixes throughout the experiment.

Table 3: Routing period for different cache size

Cache table size	*Routing period (s)*	*Cache table size*	*Routing period (s)*
5	1866	500	1306
10	1883	700	1207
50	1751	800	1161
100	1660	900	1144
150	1589	1000	1089
300	1440	10000	807

Table 3 give the measured lookup time, however, We can see from the result that increases the number of entries in the cache routing table, the search time to forward IP packets is reduced as shown in Figure 2.

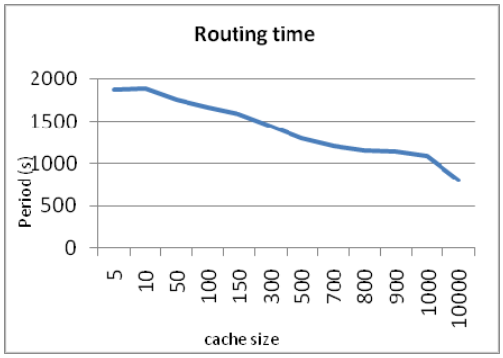

Fig.2. Graphical representation of table 3 results

In a second test, we evaluate the performance of our proposed algorithm; we used various sizes of routing tables generated by our tool (prefixes generator tool). We compared our algorithm with other proposed scheme in terms of the search time.

Table 4 shows the performance comparison of the proposed algorithm with an algorithm uses a binary trie [3] for represents a routing table, we used a main routing table with 10000 prefixes and a cache routing table with 0.1% of the main routing table size. The proposed algorithm is the best in the average time search. Since the search process in our proposed algorithm stops when a next hop is found in a cache table,

Table 4: Comparing our algorithm with the binary trie based search algorithm

IP packets series	*Routing with cache (s)*	*Routing with binary trie (s)[3]*
2345	100	375
22349	500	5028
226588	850	1978
450700	3939	12000

The figure 3 shows the distribution of search time for the four sets of IP packets tested.
It is clear in the figure 3 that our algorithm improves the search time for all series of IP packets tested

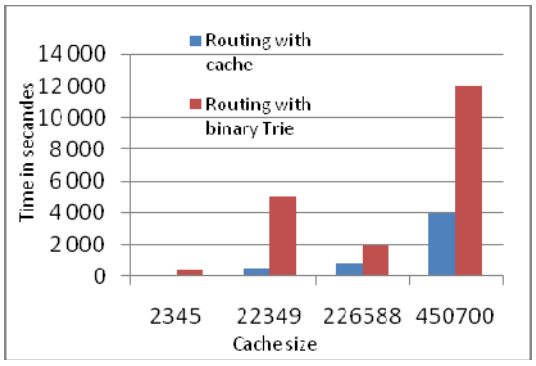

Fig.3. Time search distribution for the IP packets sets tested

7. Conclusion

In this paper, we have proposed an algorithm for the best address lookup operation that uses the cache routing table approach. The IP routing table is structured in main routing table and cache table, the cache routing table is able to store the IP addresses recently looked, in the proposed cache table; we not only cache the prefix destination address of the packet but also cache the prefix for used in the routing table updates.
Our main contribution was to propose an algorithm to speed the IP address lookup operation. To demonstrate the effectiveness of our algorithm, comparisons are made with the binary trie based algorithm [3], the simulation results show that our approach offers better search speed performance compared with the binary trie based packet forwarding algorithm.

References

[1] Haibin Lu and Sartaj Sahni "O (log n) Dynamic Router-Tables for Prefixes and Ranges" IEEE transactions on computers, vol. 53, no. 10, October 2004
[2] S. Nilsson, G. Karlsson," IP address lookup using LC-tries", IEEE Journal on Selected Areas in Communications, vol 17, issue 6, pp 1083–1092, 1999.
[3] R.C. Chang and B.-H. Lim " Efficient IP routing table lookup scheme" IEE Proc. Commun., Vol. 149, No. 2, pp.77-82, April 2002
[4] Chi-Cheng Wu Jia-Long Wu Huang-Sen Chiu " Improving IP lookup Performance with a Routing-Table Cache" IEEE Communications Letters, Volume 7, Issue 3, March 2003
[5] Chang, R.C. Beng-Huat Lim "Efficient IP routing table VLSI design for multigigabit routers " IEEE International Symposium on vol 2,pp. 776-779 2002
[6] Jinsoo Kim and Junghwan Kim "An Efficient IP Lookup Architecture with Fast Update Using Single-Match TCAMs" Springer-Verlag Berlin Heidelberg, pp. 104–114, 2008
[7] M. Ruiz-Sánchez, E. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms", IEEE Network Magazine, March/April 2001.
[8] David Antos "Overview of Data Structures in IP Lookups " CESNET Tchnical Report 9/2002.
[9] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy" RFC 1519, September 1993
[10] Anindya Basu Girija Narlikar "Fast Incremental Updates for Pipelined Forwarding Engines" IEEE/ACM Transactions on Networking (TON), Volume 13 , Issue 3, pp. 690 – 703, June 2005
[11] P. Gupta, S. Lin, and N. McKeown. "Routing Lookups in Hardware at Memory Access Speeds". In Proceedings of Infocom '98, pp. 1240–1247, San Francisco, April 1998
[12] Mohammad J. Akhbarizadeh, Mehrdad Nourani " Hardware-Based IP Routing Using Partitioned Lookup Table" IEEE/ACM transactions on networking, vol. 13, no. 4,pp. 769 - 781 august 2005
[13] Houassi Hichem, Bilami Azeddine "Une architecture parallèle pour la consultation des tables de routage "The 5th international Conference: sciences of electronics, technologies of information and telecommunication SETIT 2009, Tunisia , March 2009.
[14] M. J. Akhbarizadeh and M. Nourani "Efficient prefix cache for network processors" In 12th Annual IEEE Symposium on High Performance Interconnects, pp. 41–46, Aug 2004.
[15] S. Kasnavi, P. Berube, V. Gaudet, and J.N Amaral "A cache-based Internet Protocol address lookup architecture" IEEE, Computer Networks, Vol. 52, Issue 2, pp. 303-3268, February 2008.
[16] Soraya Kasnavi, Paul Berube, Vincent C. Gaudet, and Jos Nelson Amaral "A Multizone Pipelined Cache for IP Routing " LNCS 3462, pp. 574–585, networking 2005.
[17] Kaushik Rajan and Ramaswamy Govindarajan, Senior Member , "A Novel Cache Architecture and Placement Framework for Packet Forwarding Engines", IEEE Transactions on computers, VOL. 58, NO. 8, pp.1009-10025, AUGUST 2009

[18] L. Peng, W. Lu, and L. Duan " Power efficient IP lookup with supernode caching" IEEE GLOBECOM, Vol. 48, pp. 215–219, November 2007.
[19] L.C Wuu, T.J Liu, and K.M Chen "A longest prefix first search tree for IP lookup" Elsevier,Computer Networks, Volume 51, Issue 12, pp 3354-3367, August 2007
[20] N. Yazdani and P. S. Min, "Fast and scalable schemes for the IP address lookup problem," in Proc. IEEE HPSR2000, pp 83-92, 2000.
[21] H. Lim and J. Mun, "An efficient IP address lookup algorithm using a priority-trie," IEEE Globecom, pp. 1-5, Nov. 2006.

Houassi Hichem received the magister degree in 2004 from the Department of Computer Science, University of Batna, Algeria. He is currently an assistant professor at the University Center of Khenchela, Algeria. He is currently a Doctoral student in the Department of Computer Science, University of Batna, Algeria. His research interests include router architectures, IP-address lookup algorithms, and mobile networks.

Bilami Azeddine is professor in the Department of Computer Science, University of Batna, Algeria. His research interests include Wireless Networks, network security and router architectures.