IJCTA, 9(5), 2016, pp. 2461-2465

Trie Network Lookup

Joshua Ranjit Christian M.[1], **Sujatha V.**[2], **Jayanthi S.**[3],
Akshay R.[4] **and Divyush Raj A.**[5]

ABSTRACT

Networks and routing have become an integral part of life on earth. Networked computing works on routing the packets aroundthenetwork.Theexperimentistoproposean ew algorithm to reduce the time taken to find the next hop in the lookup table by implementing a partitioned and encoded search mechanism which reduces the time taken from n to square root of n in look-up. This is done by using a trie data structure for the network addressed encoded in Huffman codes. The class-wise multithreading also help s in achieving faster results. The new algorithm is to improve the timeefficiency.

Keywords: Look-up table, Network address, Next hop, Trie data structure, Multi-threading, Huffman Code, Router.

1. INTRODUCTION

Computer networks is an important aspect in our daily progress as we depend on networked computers for almost everything nowadays. The primary role of routers is to forward packets towards their final destination. To this purpose, a router must decide for each incoming packet where to send it next. the next-hop router as well as the egress port through which the packet should be sent. This forwarding information is stored in a forwarding table that the router computes based on the information gathered by routing protocols. To consult the forwarding table, the router uses the packet's destination address as a key; this operation is called address lookup. Once the forwarding information is retrieved, the router can transfer the packet from the incoming link to the appropriate outgoing link, in a process called switching. The address lookup operation is a major bottleneck in the forwarding performance of today's routers. This paper presents a technique to improve the efficiency of the existing look up algorithm.

2. EXPLANATION

In IP version4, IP addresses are 32 bit long and, when broken up into 4 groups of 8 bits, are normally represented as four decimal numbers separated by dots. For example, the address 10000010_01010110_00010000_ 01000010 corresponds in the dotted-decimal notationto130.86.16.66. IP addresses use a two level hierarchy, with network layer on top and the host layer at the bottom. Routing protocols mostly focus on the network part of the addresses which is called the address prefix a sit forms the first few bits of the IP address. With a two-level hierarchy, IP routers forwarded packets based only on the network part, until packets reached the destination network. As a result, a forwarding table only needed to store a single entry to forward packets to all the hosts attached to the same network. This technique is called address aggregationand allows using prefixes to represent a group of addresses. Each entry in a forwarding table contains a prefix, as can be seen in Table 1. So, finding the forwarding information requires to search for the prefix in the forwarding table that matches the corresponding bits of the destination address.

[1]&[5] Department of Computer Science and Engineering, KCG College of Technology, Chennai-600097, *Emails: jchris1901@gmail.com, sujatha@kcgcollege.com, jayanthi.jagadeesa@gmail.com, akshayrameshbabu@gmail.com, anithvinitha@gmail.com*

Table 1
A forwarding table

Destination Address Prefix	*Next-hop*	*Output interface*
24.40.32/20	192.41.177.148	2
130.86/16	192.41.177.181	6
208.12.16/20	192.41.177.241	4
208.12.21/24	192.41.177.196	1
167.24.103/24	192.41.177.3	4

The addressing architecture specifies how the allocation of addresses is performed, that is it define show to partition the total IP address space of 232 addresses. Specifically, how many network addresses will be allowed and of what size each of them should be. When the Internet addressing was initially designed, a rather simple address allocation scheme was defined, which is known today as the classful addressing scheme. Basically, three different sizes of networks were defined in this scheme, identified by a class name: class A, B, and C (see figure 1). Size of networks was determined by the number of bits used to represent the network part and the host part. Thus networks of class A, B or C consisted in an 8, 16 or 24-bitnetwork part and a corresponding 24,16 or 8-bithost part.

With this scheme there were very few class A networks and their addressing space represented 50% of the total IPv4 address space (231 addresses out of a total of 232). There were 16,384 (214) class B networks with a maximum of 65,534 hosts per network and 2,097,152 (2 21) class C networks with up to 256 hosts. This allocation scheme worked well in the early days of the Internet. However, the continuous growth of the number of hosts and networks have made apparent two problems with the classful addressing architecture. First, with only three different network sizes to choose, the address space was not used efficiently and the IP address space was getting exhausted very rapidly, even though only a small fraction of the addresses allocated were actually in use. Second, although the state information stored in the forwarding tables did not grow in proportion to the number of hosts, it still grew in proportion to the number of networks. This was especially important in the backbone routers, which must maintain an entry in the forwarding table for every allocated network address. As a result, the forwarding tables in the backbone routers were growing very rapidly. The growth of the forwarding tables resulted in higher lookup times and higher memory requirements in the routers and threatened to impact their forwardingcapacity.

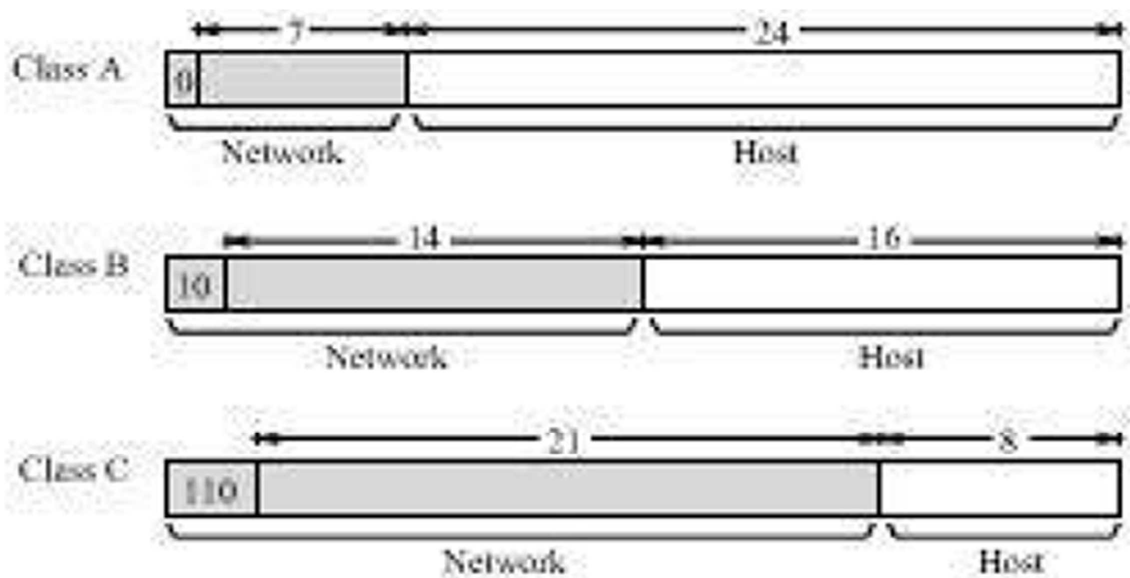

Figure 1: Classful Addresses

To allow for a more efficient use of the IP address space and to slow down the growth of the back bone forwarding tables, a new scheme called Classless Inter-domain Routing or CIDR was introduced. Remember, that in the classful address scheme, only 3 different prefix lengths are allowed: 8,16 and 24 corresponding to the classes A, B and C, respectively (see figure 1).CIDR makes more efficient use of the IP address space by allowing a finer granularity in the prefix lengths. With CIDR, prefixes can be of arbitrary length rather than constraining them to be 8, 16 or 24 bits long. To address the problem of forwarding table explosion, CIDR allows address aggregation at several levels. The idea is that the allocation of addresses hasa topological significance. Then, we can recursively aggregate addresses at various points within the hierarchy of theInternet's topology. As a result, back bone routers maintain forwarding information not at the network level but at the level of arbitrary aggregates of networks. Thus, recursive address aggregation reduces the number of entries in the forwarding table of backbone routers

In the classful addressing architecture, the length of the prefixes was coded in the most significant bits of an IP address (see figure 1), and the address lookup was a relatively simple operation: Prefixes in the forwarding table were organized in three separate tables, one for each of the three allowed lengths. The lookup operation amounted to find an exact prefix match in the appropriate table. The search for an exact match could be performed using standard algorithms based on hashing or binary search. While CIDR allows to reduce the size of the forwarding tables, the address lookup problem now becomes more complex. With CIDR, the destination prefixes in the forwarding tables have arbitrary lengths and do not correspond any more to the network part since they are the result of an arbitrary number of network aggregations.

Therefore, when using CIDR, the search in a forwarding table can not be performed any longer by exact matching because the length of the prefix cannot be derived from the address itself. As a result, determining the longest matching prefix involves not only to compare the bit pattern itself but also to find the appropriate length. Therefore, we talk about searching in two dimensions, the value dimension and the length dimension. The search methods we will review try to reduce the search space at each step in both of these dimensions. In what follows we will use N to denote the number of prefixes in a forwarding table and W to indicate the maximum length of prefixes, which is typically also the length of the IP addresses.

3. EXPERIMENT

The proposed experiment suggests that the binary code of the network address is encoded in Huffman codes and then implemented using Trie data structure. Trie is a tree based data structure which can either be a single bit trie or a multibit trie. To improve the efficiency, the trie is multithreaded under the first Octant of the IP addressing scheme. The linear search of the longest matching prefix takes a long time equal to the number of elements in the lookup table. Instead when we multithread it the complexity reduced to the

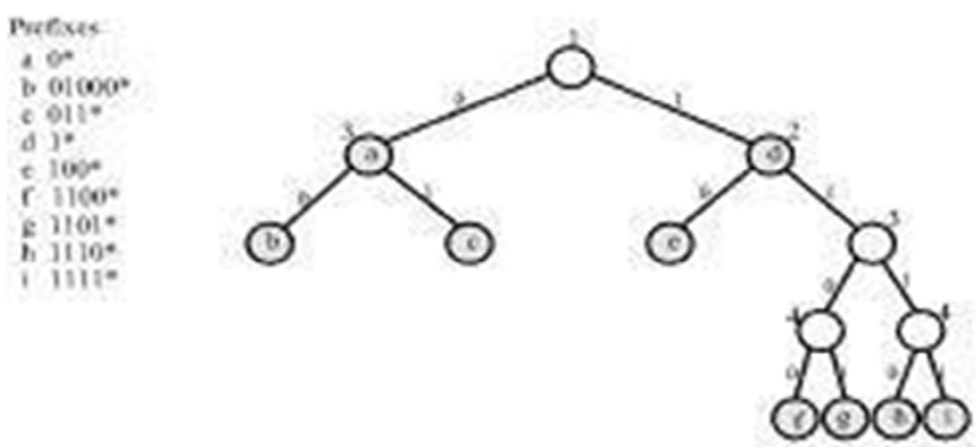

Figure 2: Path Compressed Trie

square root of the number of entries in the look up table. The encoding and decoding when done in Huffman codes can make sure that memory was tage is reduced. The usage of multi-threaded search algorithms need better hardware so cost of production will be higher than usual but time efficiency will be consider ably increased.

The multi-threaded algorithm is as follows

a) Split the IP address into three parts, first part being the first octant, second part being the rest of the IP address and the last part being the host address.

b) Neglect the host address part and consider only the second part.

c) Encode the second part in Huffman codes and construct binary tries with the next hops appended on the node.

This considerably improves the efficiency of the search and also the memory complexity of the routing mechanisms.

4. RESULT

Routing in networks is a very important process. With the increasing network addresses and number of hosts the time taken to route a packet is crucial. The proposed algorithm when implemented has proven to reduce the time complexity of routing algorithms from O(n) to O (d) by multithreading and encoding in Huffman codes.

REFERENCES:

[1] G. Cheung and S. McCanne, "Optimal Routing Table Design for IP Address Lookups Under Memory Constraints," Proceedings of IEEE INFOCOM'99, pp. 1437-1444, March 1999.

[2] P. Crescenzi, L. Dardini, R. Grossi, "IP Address Lookup Made Fast and Simple," Algorithms ESA'99, also as Technical Report TR-99-01 Università di Pisa.

[3] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, "Small Forwarding Tables for Fast Routing Lookups," Proceedings of ACM SIGCOMM'97, pp. 3-14, September 1997.

[4] W. Eatherton, "Full Tree Bit Map," Master's Thesis, Washington University, 1999.

[5] A. Feldmann and S. Muthukrishnan, "Tradeoffs for Packet Classification," Proceedings of IEEE INFOCOM 2000, pp 1193-1202, March 2000.

[6] P. Gupta, S. Lin, N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds," in Proceedings of IEEE INFOCOM'98, pp. 1240-1247, April 1998.

[7] G. Huston, "Tracking the Internet's BGP Table," presentation slides available at: http://www.telstra.net/gih/prestns/ietf/bgpt able.pdf, December 2000.

[8] C. Labovitz, "Scalability of the Internet Backbone Routing Infrastructure," Ph. D. Thesis, University of Michigan, 1999.

[9] B. Lampson, V. Srinivasan, and G. Varghese, "IP Lookups using Multi-way and Multicolumn Search," in Proceedings of IEEE INFOCOM'98, pp. 1248-1256, April 1998.

[10] D. R. Morrison, "PATRICIA-Practical Algorithm to Retrieve Information Coded in Alphanumeric," Journal of the ACM, 15(4), pp. 514-534, October 1968.

[11] S. Nilsson and G. Karlsson "IP-Address Lookup Using LC-Tries," IEEE Journal on Selected Areas in Communications June 1999, Vol. 17, Number 6, pp. 1083-1092.

[12] "Un mécanisme optimisé de recherche de route IP," in Proceedings of CFIP 2000 (Coloque Francophone sur l'Ingénierie des Protocoles), pp. 217-232. October 2000.

[13] K. Sklower, "A tree-based packet routing table for Berkeley Unix," Proceedings of the 1991Winter Usenix Conference, pp. 93-99, 1991.

[14] V. Srinivasan and G. Varghese, "Fast Address Lookups using Controlled Prefix Expansion," Proceedings of ACM Sigmetrics'98, pp. 1-11, June 1998.

[15] W. Richard Stevens and Gary R. Wright. TCP/IP Illustrated, Vol. 2 The Implementation. Addison-Wesley, 1995.du/ipma/routing_table/,16 August, 1999.

[16] Warkhede, “Multiway Range Trees: Scalable IP Lookup with Fast Updates,” Technical Report 99-28, Washington University, 1999.

[17] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, “Scalable High Speed IP Routing Lookups,” Proceedings of ACM SIGCOMM’97, pp. 25-36, September 1997.

[18] Prefix database Mae East, The Internet Performance Measurement and Analysis (IPMA) project, data available at: http://www.merit.e