A SURVEY ON THE IP ADDRESS LOOKUP TECHNIQUES

[1]Sushma G, [2]Priya D

[1]Student, [2]Assistant Professor

[1]Information Science and Engineering Department, [2]Information Science and Engineering Department,

[1]RV College of Engineering, Bangalore, India, [2]RV College of Engineering, Bangalore, India

Abstract : Routers perform many tasks like switching, scheduling, buffering of packets etc. The most important among them is the IP address lookup. As there is a rapid growth in the internet, fast IP address lookup is crucial. Therefore, various existing efficient approaches for IP address lookup have been summarized in this paper.

***IndexTerms* - Address lookup, longest prefix matching, binary tree, trie, binary trie.**

I. INTRODUCTION

The important task of a router is to check the final destination IP address of a packet and forward it to that address. Many existing efficient approaches employ the well known longest matching prefix approach. This is to identify the item which should be utilised for a particular packet address in the routing table. It involves checking the forwarding or routing table for the longest prefix that fits the incoming packet's destination address. Many architectures, algorithms and approaches like hashing based schemes, trie based schemes, binary search schemes etc have been proposed which have been summarized in this paper.

II. SURVEY

[1] For the purpose of looking up the IP address in software faster, the data structure LC-trie is proposed. The performance is very high, upto 1 million IP addresses can be processed on a pentium 133 Mhz computer. The fundamental idea of this paper is that comparing different strings which have identical lengths or lengths which is less than a word (typically the largest integer in a single instruction the computer can process) is essentially a constant cost. Thus it's better to compare more bits at once to limit the memory accesses the system has to perform and the number of comparisons it has to do. By constructing IP-routing tables as level-compressed tries, it is proven how they may be concisely described and effectively searched. Generis data structure is used which is not based on any arbitrary preconceptions about the prefix length distribution in routing tables. Merely that an address is a binary string is assumed. Despite the fact that the data structure makes no explicit presumptions regarding address distribution, it adjusts nicely: The sparse sections of the trie are compressed via path compression, whereas the dense sections are compressed by level compression.

[2] Assess the performance of modern IP address lookup algorithms concerning scalability, speed of lookup and overhead of update. Describes the addressing architecture like classful addressing scheme, CIDR addressing scheme which is important to routing architecture. They Illustrate some of the classical solutions like a binary trie approach, path-compressed tries. Using multibit tries, the search on prefix lengths is described. Report that exact matching is more simple than longest prefix matching. In order to enable speedier search, the lookup techniques examined in this paper alter prefixes by performing regulated disaggregation. Because original prefixes are typically converted into multiple prefixes, adding, deleting, or modifying a single prefix necessitates updating several entries, but in the worst scenario, the complete data structure must be reconstructed. As a result, lookup time and incremental update time trade-off must be considered in general. A framework is given and categorises the schemes based on the algorithm-data structure view.

[3] Suggests an optimal binary trie structure which is called priority trie to solve the IP address lookup problem utilizing range representation of prefixes. On a number line, without growing to the maximum length, the prefixes are expressed as ranges within 0 and 1 in this range format. The lowest range that includes the input address is the best match for a particular input address. The priority trie is built by taking trie structure as the base for design, with the priority prefix that is the longest with many in the subtrie rooted with the empty nodes replacing the trie's empty internal nodes. When an input matches a priority prefix, the search is completed, considerably improving the performance of search. The suggested priority trie performs

exceptionally well in performance parameters such as scalability, memory size, lookup speed, update performance as demonstrated by real-world routing data.

[4] SuggestS that the absence of a structured approach for comparing and sorting strings of varied lengths when some are prefixes of others, is the basic issue impeding the application of common tree architectures to the IP lookup problem such as m-way trees. As a result, the first step is to design a simple mechanism for comparing and sorting IP address prefixes. Later, for applying the familiar data structures to the IP lookup problem, the prefix data is changed and is tuned. A binary prefix tree and methods for sorting and constructing the index structure. It works well for the problem of longest prefix matching because it utilises memory space of O(n). Static and dynamic m-way prefix trees are offered as two types of m-way trees.

[5] Presents an efficient binary search technique for IP address lookup. The suggested binary tree's depth is very near to the minimal bound, so when compared to earlier techniques, it leads to a significantly lower count of worst case memory accesses. The suggested approach can be employed in realistic routers for software based lookup of address and uses a modest amount of memory. The data from real-world backbone routers with varying numbers of prefixes is obtained and tested with the suggested binary tree. When this was compared to the tree obtained in [4], the suggested binary tree has a 0.4 to 0.8 percent more computational time overhead.

[6] Inspected a realistic IP address lookup strategy in which the problem of longest prefix matching is converted to the exact matching problem. The forwarding table in the proposed design is made up of numerous SRAMs, all of which reflect a single prefix address lookup table.

Every address lookup table is subjected to hashing methods for locating similar entries in parallel, with the entry matching the longest prefix being chosen. A huge routing table having 37,000 entries is compressed to a 189 kb forwarding table and for each of two memory accesses, one route lookup on average is achieved, according to experiments utilizing MAE-WEST router's data. The suggested technique is an outstanding hardware architecture that conducts the process of address lookups with a modest quantity of memory and a fair number of access to memory.

[7] Offset Encoded Trie (OET) is a unique offset encoding scheme for IP address lookup which is a memory-efficient approach proposed in this work.Without the pointers for next hop and the child pointers, OET's every node carries simply an offset value and a next hop. To calculate the address of the next node to be searched, every node of traversal utilizes the offset value and bitmap of next hop as two offsets. The proposed OET uses much less memory when compared to earlier multi-bit trie systems, according to the performance study on real-world IP prefix tables. Testing on real IPv6 and IPv4 prefix tables, for example, demonstrates that the OET saves 55 to 63 percent and 60 to 76 percent of memory, respectively when compared to the Tree Bitmap Trie.

[8] A bloom filter which is onchip determines a node's membership in an offchip trie in previous proposed techniques, reducing the number of access to trie since the accesses to nodes which are not existing in the trie could be filtered out by Bloom filter. For the IP address lookup problem, a new method is proposed by using Bloom filter. Experiments convey that the best prefix which matches can be determined on average by with only one off-chip access and with a Bloom filter of appropriate size in the worst case in this method.

[9] Proposes a novel IP address lookup strategy which is based on prefix length binary search.The suggested technique employs leaf-pushing instead of the prior binary search approach on prefix length, which involves extensive computations done initially of markers and the prefixes that best match in the inner nodes. In leaf-pushed trees, primitive binary search is conducted on prefix length.It achieves excellent results while using an acceptable amount of memory, in terms of memory access count required for an address lookup. The results obtained from the performance study also reveal that the suggested system performs well in terms of scalability and speed of lookup.

[10] Suggests a trie structure based algorithm. But here the priority prefixes substitute internal nodes which are empty. As the search may be completed instantly after the priority prefix is matched with the input, the suggested algorithm's longest prefix matching is more efficient. This priority trie performs admirably in terms of lookup speed, scalability and the memory requirement according to the findings of the evaluation for performance.

III. CONCLUSION

As IP address lookup is the most important function that a router has to perform, it is crucial to use the most efficient approach. This paper is a summary of the many approaches used for IP address lookup.

REFERENCES

[1] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," in IEEE Journal on Selected Areas in Communications, vol. 17, no. 6, pp. 1083-1092, June 1999

[2] M. A. Ruiz-Sanchez, E. W. Biersack and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," in IEEE Network, vol. 15, no. 2, pp. 8-23, March-April 2001

[3] H. Lim, C. Yim and E. E. Swartzlander, "Priority Tries for IP Address Lookup," in IEEE Transactions on Computers, vol. 59, no. 6, pp. 784-794, June 2010

[4] N. Yazdani and P. S. Min, "Fast and scalable schemes for the IP address lookup problem," ATM 2000. Proceedings of the IEEE Conference on High Performance Switching and Routing (Cat. No.00TH8485), 2000, pp. 83-92.

[5] Changhoon Yim, Bomi Lee and Hyesook Lim, "Efficient binary search for IP address lookup," in IEEE Communications Letters, vol. 9, no. 7, pp. 652-654, July 2005.

[6] Hyesook Lim, Ji-Hyun Seo and Yeo-Jin Jung, "High speed IP address lookup architecture using hashing," in IEEE Communications Letters, vol. 7, no. 10, pp. 502-504, Oct. 2003.

[7] K. Huang, G. Xie, Y. Li and A. X. Liu, "Offset addressing approach to memory-efficient IP address lookup," 2011 Proceedings IEEE INFOCOM, 2011, pp. 306-310.

[8] J. H. Mun and H. Lim, "New Approach for Efficient IP Address Lookup Using a Bloom Filter in Trie-Based Algorithms," in IEEE Transactions on Computers, vol. 65, no. 5, pp. 1558-1565, 1 May 2016.

[9] Ju Hyoung Mun, Hyesook Lim and Changhoon Yim, "Binary search on prefix lengths for IP address lookup," in IEEE Communications Letters, vol. 10, no. 6, pp. 492-494, June 2006.

[10] H. Lim and J. H. Mun, "NXG06-1: An Efficient IP Address Lookup Algorithm Using a Priority Trie," IEEE Globecom 2006, 2006, pp. 1-5.

[11] Xiaojun Nie, D. J. Wilson, J. Cornet, D. Damm and Yiqiang Zhao, "IP address lookup using a dynamic hash function," Canadian Conference on Electrical and Computer Engineering, 2005., 2005, pp. 1642-1647.

[12] X. Sun and Y. Q. Zhao, "An on-chip IP address lookup algorithm," in IEEE Transactions on Computers, vol. 54, no. 7, pp. 873-885, July 2005.

[13] D. Pao, C. Liu, A. Wu, L. Yeung and K. S. Chan, "Efficient hardware architecture for fast IP address lookup," Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 2002, pp. 555-561 vol.2.

[14] H. Fadishei, M. S. Zamani and M. Sabaei, "A novel reconfigurable hardware architecture for IP address lookup," 2005 Symposium on Architectures for Networking and Communications Systems (ANCS), 2005, pp. 81-90.

[15] Fu, Jing & Rexford, Jennifer. “Efficient IP-address lookup with a shared forwarding table for multiple virtual routers”.(2008)