A Hybrid Binary Search Scheme for IP Address Lookup

Hyuntae Park **Hyun-Sik Kim** **Sungho Kang**
Dept. of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
radiob@soc.yonsei.ac.kr hskim@soc.yonsei.ac.kr shkang@yonsei.ac.kr

Abstract – *In this paper, a new binary search scheme for IP address lookup is proposed. There are problems of a binary search for IP address lookup. The problems have mainly attributed to an enclosure relationship of prefixes. In the proposed hybrid binary search scheme (HBS), all prefixes are classified into a disjoint set and an enclosure set. Then, the appropriate binary search schemes for each set are adopted. The disjoint set and the enclosure set are stored in separate forwarding tables, and searched in parallel. Accordingly, our proposed hybrid scheme can perform perfectly balanced binary search for IP address lookup. The performance evaluation results show that the proposed scheme has very good performance in terms of the lookup speed and the required memory size.*

Keywords: IP address lookup, binary search, binary prefix tree, binary search on range, disjoint prefix, enclosure prefix

1 Introduction

Thanks to the deployment of optical fiber technologies, broader link bandwidth in higher speed can be served to network customers. Accordingly, the packet forwarding on wire speed becomes more critical in the cross point of network as routers. One major function in packet forwarding is to look up the destination address of an incoming packet in the forwarding table, called IP address lookup. In order to allow for arbitrary aggregation of network addresses, classless inter-domain routing (CIDR) is deployed instead of class-based addressing. It is necessary to find the most specific match among matching entries with various lengths, called the longest matching prefix (LPM). Therefore, we need to search among the space of all prefix lengths, as well as the space of all prefixes of a given length. In other words, exact matching for class-based addressing is the search of one dimension through prefix data, whereas LPM for CIDR is the search of two dimensions through prefix lengths and prefix data. Obviously, this complicated task is a critical issue for packet forwarding on wire speed [1].

2 Previous Works

Binary trie [2] provides an easy way to handle prefixes with arbitrary lengths. In commercial routers, hardware-based schemes such as using TCAM [3] or using hashing [4] are commonly used. In recent years, in order to complement trie-based or hardware-based schemes, tree-based binary search schemes have attracted the attention of researchers. Representatively, binary search on range (BSR) [5] is proposed. In this scheme, prefixes represent as an interval which has a start point with padding 0 to the longest length of prefix and an end point with padding 1. The search of two dimensions through prefix lengths and prefix data for LPM can be simple to the search of one dimension through prefix data as exact matching. A match pointer and a larger pointer are precomputed and stored in each entry. Through a perfect binary search is performed on intervals of prefixes, the matched pointer becomes the best match prefix (BMP). When the search is finished, the next-hop-pointer is decided by the final BMP.

Binary prefix tree (BPT) [6] is proposed as an alternative way. An excellent contribution of this scheme is to provide the solution for sort and comparison of prefixes with various lengths. Although a perfect binary search is impossible and the tree is very deep because it is very unbalance tree, the memory efficiency is good because it has no empty internal node, no duplication and no additional pointers. BPT has continuously improved. In order to construct the balanced BPT considering the number of descendents to select an entry in each node, weighted binary prefix tree (WBPT) [7] is proposed. As well, partitioned binary prefix tree (PBT) [8] is proposed. It is built and searched on multiple BPTs by splitting into disjoint prefixes, enclosure prefixes and child prefixes.

3 Problems Definitions

There are inherent problems to apply a binary search for IP address lookup. First, all of data for a binary search must be sorted with any standard such as magnitude. Because LPM is two dimensions search for prefix data and prefix lengths, the specific standard for sorting and transforming into one dimension search must be defined. Second, although sorting is successfully done, it may be a misleading match. It is due to the enclosure relationship of prefixes. In order to avoid the misleading match, a shorter prefix must be located on the upper node than a longer prefix among enclosure prefixes. Hence, it must be

considered in the building process. Finally, although there is any matched entry, the search process cannot be finished immediately and must continue on a leaf of a prefix tree. It is due to the enclosure relationship of prefixes as well. If any prefix is matched in the middle of the tree, it is temporarily stored as the best match prefix (BMP). During the trace to a leaf of the tree sequentially, BMP must be updated if the longer prefix is matched. Therefore, a binary search for IP address lookup is absolutely kept to a leaf of a prefix tree. Ultimately, increasing the depth of a prefix tree worsen the performance in terms of a lookup speed. These problems still exist in improved BPTs such as WBPT and the enclosure tree of PBT. Consequently, an efficient binary search for LPM must be able to sort prefixes with arbitrary lengths smartly and to deal with enclosure prefixes.

There are the three approaches to reduce the required memory in the previous schemes. First, the results of pre-computation are not stored on each node of a tree. In some previous schemes, the pre-computation results must be stored in order to accelerate the search mechanism. Second, if a prefix tree is unbalanced, sub-pointers (left or right) are stored on each node for a trace as in BPT. Obviously, the memory resource for the structure of each node is increased critically. It is confirmed by our analysis for the memory utilization based on experimental comparisons. It is observed that sub-pointers or pre-computation results occupy a large portion of memory. Finally, if possible, prefix duplication must be free. In BSR and leaf-pushing based schemes, prefix duplication are created by dissolving enclosure prefixes to disjoint prefixes. These cause an increment of the number of entries. Hence, the number of nodes on the prefix tree increases tremendously by more than twice.

4 The Proposed Scheme

In this paper, we propose the hybrid binary search scheme (HBS) which solves adequately problems mentioned above. The goal of the proposed scheme is to perform a perfectly balanced binary search for IP address lookup according to the inherent characteristic of prefixes. First of all, all prefixes are classified into a disjoint set and an enclosure set in order to handle each prefix set in distinct tables.

Consider a disjoint set. Disjoint prefixes have no corresponding longer prefix because these are mutual exclusively. Through a simple sorting by a magnitude of prefixes, the perfectly balanced tree can be built. Besides, if any prefix is matched, the search operation can be finished at once. Therefore, the lookup speed of it is less than $O(\log_2 N)$ where N is the number of entries. Furthermore, the required memory is minimized because there are no sub-pointers or no pre-computation.

Consider an enclosure set. The sorting among enclosure prefixes should be handled with more complexity. In the proposed scheme, the dimension for prefix lengths is eliminated using the search based on range. Then, through a simple sorting, the perfectly balanced tree can be built as well. Besides, the amount of entries can be shrunk as the amount of disjoint prefixes by separating enclosure prefixes from disjoint prefixes. It has a feasible memory size with scalability.

4.1 Hybrid Binary Search Scheme

The proposed hybrid binary search scheme is shown in Fig. 1. It is composed of the Build & Update Unit, the Hybrid Controller, the Disjoint Search Unit and the Enclosure Search Unit with the corresponding forwarding tables. The input of the proposed scheme is the destination address of incoming packets. It is entered into the two search units concurrently in the search operation as well as the build operation. In the Disjoint Search Unit and the Enclosure Search Unit, the perfect binary search is performed with the corresponding tables. These are used for IP address lookup as well as the classification of prefixes. The classification is the same as the pre-search. The classified prefixes are passed to the Build & Update Unit by a feedback of the Hybrid Controller. Then, these are stored selectively in either the disjoint table or the enclosure table by the Build & Update Unit. The Build & Update Unit manages the forwarding tables according to the pre-search result of the Hybrid Controller. The Hybrid Controller handles the feedback of the pre-search result to the Build & Update Unit and the output of the next-hop-pointer as the search result.

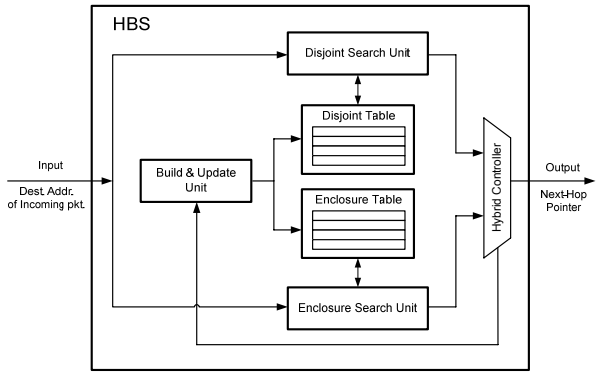

Figure 1. Proposed Hybrid Binary Search Scheme

4.2 Building

The input prefix is primarily classified into a disjoint set and an enclosure set. The classification is decided by the pre-search operation. If there is the matched entry through the pre-search operation, the input prefix and the matched entry are included in an enclosure set. If not, the input prefix is added in a disjoint set. For the given

sample set of prefixes, classified prefixes are shown in Fig. 2. According to the type of prefixes, the disjoint table and the enclosure table are composed of the corresponding prefix set separately by the Build & Update Unit. In the disjoint table, the perfectly balanced binary prefix tree is built by just sorting because a disjoint set is exclusive. All of disjoint prefixes are sorted by referring magnitude comparison of various length proposed in the original BPT. Based on a sample set of Fig. 2, the binary prefix tree and the table for a disjoint set are shown in Fig. 3. In the enclosure table, prefixes are padded by 0 as a start point and padded by 1 as an end point. It can be sorted justly because all of padded entries have the same length. After sorting, the larger pointer(>) and the equal pointer(=) are precomputed. The pre-computation of pointers is the same as it of the original BSR. Sorted and padded prefix with the corresponding pointers are stored in the enclosure table shown in Fig. 4.

4.3 Searching

The destination address of incoming packets is searched by the dedicated search units in parallel. A perfect binary search is performed based on both the disjoint and the enclosure table. If the matched prefix is found in the disjoint table or in the enclosure table with the equal pointer, the search is immediately completed and the output is decided by the corresponding next-hop-pointer. In the enclosure table, the binary search is continued until there is no more entry for comparison. In the middle of the search trace, whenever the input is larger than the compared entry, the best match prefix (BMP) is updated as the new larger pointer. When the search trace is done, if there is no matched equal pointer, the corresponding next-hop-pointer is decided as the final BMP. If there is no entirely matched prefix in both tables, the output is replaced by the default next-hop-pointer.

Let us consider an example of the input prefix 11011. In the disjoint table, it is compared to 01100, 01111, and 111 in order of precedence. But, there is no matched entry in the disjoint tree. In the enclosure table simultaneously, the input is compared with 101111 in the exact middle of the tree. As 11011 is larger than 101111, the larger pointer, P9 is stored as BMP. By the same process, the input is compared with 110010, 110011, and 110111 sequentially. Accordingly, BMP is updated as P9 after P10. 11011 is not larger than 110111 and the matched equal pointer does not exist such that the final BMP, P9 is decided to the corresponding next-hop-pointer. Finally, the output is decided by the next-hop-pointer of 110.

4.4 Incremental Updating

The proposed scheme is available for an incremental updating. The updating is the same operation as the searching. The classification of inputs is decided by the pre-search operation in the search units and the Hybrid Controller. It is allocated into the proper table in the Build & Update Unit. Therefore, an additional operation is not required for updating.

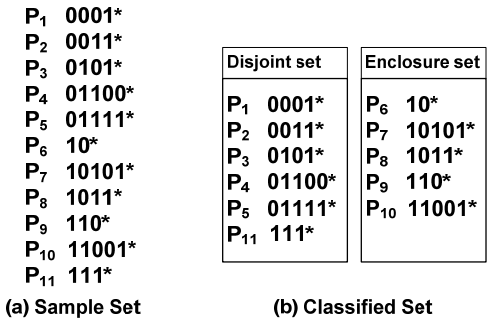

Figure 2. An Example Set

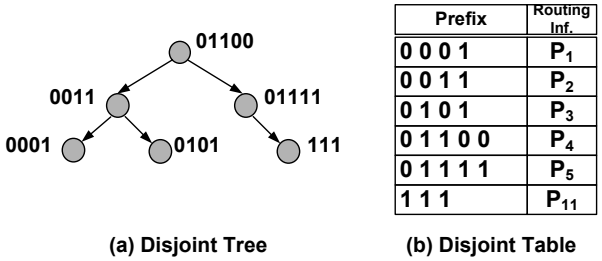

Figure 3. Disjoint Tree and Table for Binary Prefix Tree

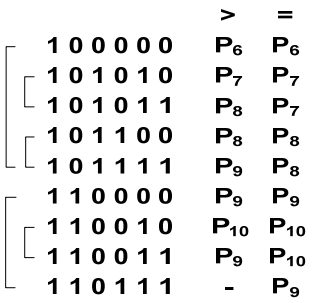

Figure 4. Enclosure Table for Binary Search on Range

5 Performance Analysis

The performance of the proposed scheme is evaluated using various routing data from real backbone routers in order to consider a variety of IP address environments. Mae-West1 and FUNET include a lot of disjoint prefixes and have about 30~40K entries, it is fewer amount of entries relatively. On the other hand, PORT80 and Telstra include enormous enclosure prefixes and have about 100~250K entries, it is larger amount of entries. It has more depth of hierarchical IP aggregation. Generally, whereas the level of IP aggregation is 4 or 5 as

in Mae-West1 and FUNET, that is 8 or 9 in PORT80 and Telstra.

Fig. 5 and Fig. 6 show the performance comparison with binary trie[2], binary search on range[5], BPT[6], WBPT[7], and PBT[8] in terms of the average number and the maximum number of memory accesses. In Mae-West1 and FUNET, the proposed scheme is improved remarkably. On the other hand, in PORT80 and Telstra, the performance is slightly better than BSR. The whole performance is overwhelmed by the search process for enclosure prefixes because the search of enclosure prefixes is more complex than it of disjoint prefixes. In any case, the proposed scheme has the best performance in terms of the lookup speed in both the average and the worst cases. Besides, in the proposed scheme, the required memory size is reduced as shown in Fig. 7. Although PORT80 and Telstra have a lot of the entries and the deep level of the IP aggregation, unnecessary pointers of the disjoint set can be reduced according to the classification of a disjoint set and an enclosure set. Therefore, the proposed HBS shows better performance than BSR and BPT in terms of the required memory.

Over all, in the performance of the proposed HBS, the lookup speed is the same as BSR, and the requited memory is the same as PBT. The required memory size is 0.86Mbytes about 112K entries of PORT80 router. An address lookup is achieved by 16.89 memory accesses in average and by 19 memory accesses in the worst case.

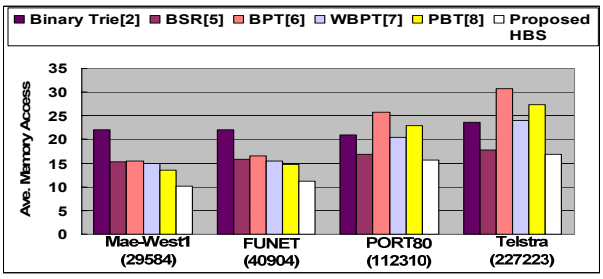

Figure 5. Comparison of the Average number of Memory Accesses

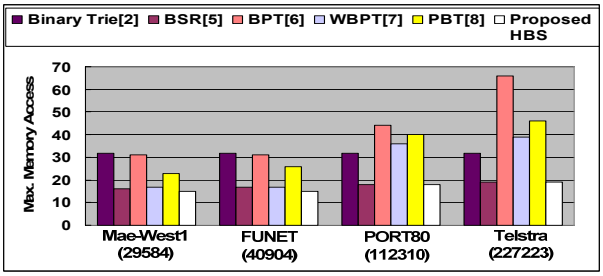

Figure 6. Comparison of the Maximum number of Memory Accesses

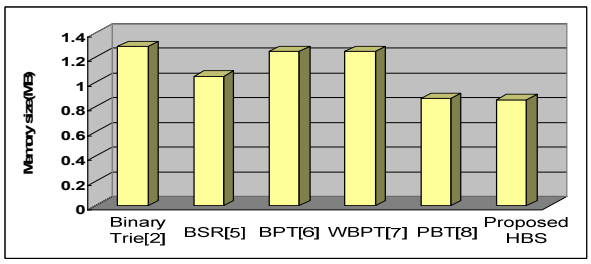

Figure 7. Comparison of the Required Memory Size(MB)

6 Conclusions

In this paper, the hybrid binary search scheme for IP address lookup is proposed. After all prefixes are classified into a disjoint set and an enclosure set, the appropriate binary search schemes for each set are adopted. Accordingly, the proposed scheme can be performed the perfectly balanced binary search in parallel. The performance analysis results show that the lookup speed is improved and the required memory size is reduced.

References

[1] H. J. Chao, "Next generation routers," *Proceedings of the IEEE*, vol. 90, pp. 1518-1558, Sept. 2002.

[2] M. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, pp. 8-23, Mar./Apr. 2001.

[3] D. Shah, and P. Gupta, "Fast updating algorithms for TCAM," *IEEE Micro*, vol. 21, Issue 1, pp. 36-47, Jan.-Feb. 2001.

[4] Y. Chu, P. Lin, J. Lin, H. Su and M. Chen, "ASIC Design of Fast IP Lookup for Next Generation IP Router," in proc. of ISCAS2005, pp.3825-3828, May 2005.

[5] 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.

[6] N. Yazdani and P. S. Min, "Fast and scalable schemes for the IP address lookup problem," in Proc. IEEE HPSR, pp. 83-92, 2000.

[7] C. Yim, B. Lee, and H. Lim, "Efficient binary search for IP address lookup," *IEEE Communication Letter*, vol. 9, pp. 652-654, July 2005.

[8] H. Park, B. Moon, and S. Kang, "A partitioned binary search scheme on multiple trees for efficient IP address lookup," in Proc. ISOCC, pp. 47-50, Oct. 2006.