Enhanced Patricia Tree with Reordering and Fast Incremental and Decremental Update Functions

Shin-ichi Ishida, Koji Ikehara, and Hiroaki Nishi

***Abstract*—Routing-table lookup function is formed as a tree structure. In particular, backbone router of the Internet manages a huge tree structure of the routing table in order to store all routing information brought by routing protocols, such as BGP. Though high-speed lookup is desirable for the backbone router, the tree-based lookup requires multiple memory accesses and it degrades the throughput of the lookup. Aggregation of redundant nodes and elimination of useless nodes are necessary for the high-throughput lookup. Enhanced Patricia Tree (EPT) and Enhanced Patricia Tree with Reordering (EPT-R) were studied for the high-throughput lookup. In this paper, the incremental and decremental update algorithms for EPT-R are proposed and evaluated. These incremental and decremental update algorithms for EPT and EPT-R are effective in quick update time.**

***Index Terms*—Enhanced patricia tree with reordering, internet router, routing lookup.**

I. Introduction

Recently, the deployment of broadband networks and ubiquitous networks are enlarging the size of the Internet. This movement is accelerated by developing more attractive services. The Internet services have shifted from text-based information to audio and video based information in a short term. This movement increases the number of hosts connected to the Internet. Ubiquitous network is one of the predominant information systems that enable an "anytime and anywhere" computing environment and it will become a reason of the enlargement of the Internet in the future.

The IPv4 address space will be exhausted in couple of years and the movement of the shift from IPv4 to IPv6 will be active. At the same time, these streaming-based services requires high throughput network. IEEE802.3 standards committee includes 40Gbps and 100Gbps Ethernet Task Force (IEEE P802.3ba) to accommodate increasing Internet traffic and diversifying these services.

These advancements cause the increase the number of routes as well as nodes in the Internet. Routing table lookup is a basic function to determine the process of forwarding packets for an internet router. This routing process should be done with keeping wire-rate packet processing. Since routing table lookup is a comparatively time-consuming task, high-throughput routing table lookup, which can meet to the next generation high-throughput and extended Internet, are strongly recommended.

To address this requirement, reduced binary tree structures have been studied. EPT uses one of the reduced binary tree structures proposed by our previous study [1], [2]. EPT enables to reduce memory resources and lookup latency by using a sophisticated algorithm. EPT can attain a smaller tree compared with Patricia tree and Trie, which are believed to be optimized structures.

For further reduction of the number of routing memory accesses, we imported the reordering function into EPT. Enhanced Patricia Tree with Reordering (EPT-R) is a sophisticated high-throughput routing lookup by using localities in network traffic, and it can improve the cache-hit rate of routing cache. EPT-R uses this feature and can reduce the number of memory access. The reordering function of tree structure often increases the calculation cost of tree structure. This feature may cause some problems in the future real-time Internet services, especially for achieving a quick handover or preventing security attacks. For example, quick anti-phishing function can be implemented by updating a routing table incrementally. In this paper, EPT-R is described and the incremental and detrimental update algorithms for EPT-R are proposed

The rest of this paper is organized as follows: existing techniques of routing table lookup and problem of EPT-R are explained in section II. In this section, tree-based routing table lookup and the basis of EPT and EPT-R are described mainly. In section III and IV, incremental and decremental update algorithms are shown. In section V, the evaluation of proposed incremental and decremental update algorithms are discussed. Finally, this paper is summarized in section VI.

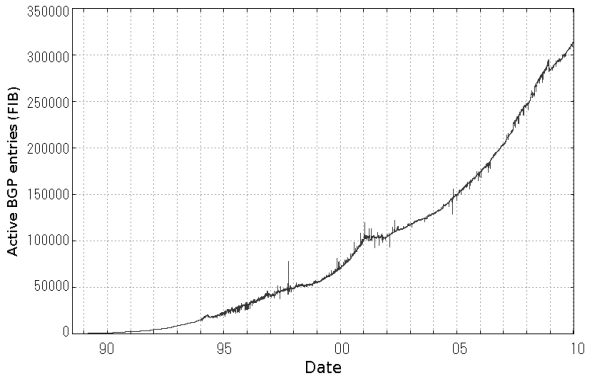


Fig. 1. The number of entries in BGP table of AS1221

II. Routing Table Lookup

Routing table is a memory that consists of IP address,

Manuscript received May 10, 2012; revised June 11, 2012. This work was supported partially supported by National Institute of Information and Communications Technology (NICT) and a Grant-in-Aid for Scientific Research (C) (22500069).

The authors are with the Graduate School of Science and Technology, Keio University, Yokohama, Kanagawa, 223-8522, Japan (e-mail: sin@west.sd.keio.ac.jp, ikehara@west.sd.keio.ac.jp, west@sd.keio.ac.jp)

DOI: 10.7763/IJIEE.2012.V2.182

prefix length, and the next-hop address. The amount of routing entries managed by a routing table is increasing in order to support growing Internet and diverse higher-level protocols or services. In routing table lookup, a rule called Longest Prefix Match (LPM) is applied. LPM is a selection rule that an entry which has the longest prefix length, namely the one with the highest subnet mask is selected in preferentially.

A. TCAM-Based Routing Table Lookup

Content Addressable Memory (CAM) is a special memory for looking up contents [3]. It has comparators at every entry and all comparators compare the value of the entry with a referencing value. CAM can achieve the lookup only in one access, because all comparators are activated at the same time. However, CAM cannot support LPM because it can consider only fixed-length data. Ternary CAM (TCAM) is a dedicated CAM for LPM search. TCAM has a similar structure with CAM and the function of the comparators is enhanced for LPM search. Each entry of TCAM has information of valid bit-length and it is used for prefix match [4]. In TCAM, each entry can detect "match or not" with taking the prefix length into consideration. Though this function is useful, there is a possibility of multiple matches. In this case, TCAM gives fastest entry in memory address as a result of lookup. Therefore, to attain LPM, entries of TCAM must be arranged in descending order of the prefix length.

High-speed routing table lookup can be attained by using TCAM. However, comparators involved in TCAM increases the total power consumption because they are activated in every lookup. Additionally, these comparators also require large hardware resources. Fig.1 shows the trend of growing BGP table [5]. The number of entries in BGP table increases like curve of the second order. In near future, there is a possibility that TCAM is not a suitable device to manage this growing routing table. In particular, future IPv6 based Internet will bring the continuous growing of the routing table. Thus, TCAM is not a suitable device to manage the problem of the growth of a routing table size in the future.

B. RAM-Based Routing Table Lookup

RAM is a memory which enables arbitrary access to the entries by using an address bus. Compared with TCAM, RAM has the merits of low-power consumption and large memory capacity. Meanwhile, RAM requires multiple memory accesses for supporting LPM. In high-speed network processor, latency of the multiple memory accesses is bottleneck of total latency. To address the bottleneck, a cache mechanism that utilizes the locality of IP traffic [6]-[8], a searching algorithm that improves access cost of memory [9]–[15] and pipeline architecture that suppress the processing latency [16]–[18] have been studied.

As a study of IP traffic locality, temporal locality of IP traffic has been studied [6]–[8]. Temporal locality means that there is a high probability of referencing the same IP address repeatedly within a short period of time. That is, an IP address that has been referenced recently is more likely to be referenced again than one that has not been referenced for a while. Cache for routing table lookup can use this locality effectively. However, cache miss causes multiple memory accesses to routing table. A searching algorithm to reduce the multiple memory accesses is required.

Most of algorithm-based solutions for reducing multiple memory accesses of routing table lookup use a tree search algorithm. The average cost of the tree search algorithm is O(log n), and is lower than the average cost of a linear search algorithm, which is O(n). Here, 'n' is the number of nodes. Tree search requires a memory access in each step. Therefore, aggregation and elimination of redundant nodes are an important factor to achieve high-throughput routing table lookup.

C. Tree-Based Routing Table

Ordinarily, routing table forms tree-based structure because the structure can provide an efficient LPM scheme for routing table lookup. However, the usual binary tree needs larger number of nodes and stages than other tree structures because the tree has redundant nodes that do not contribute to giving the answer of routing table lookup directly. In specially designed tree structure for routing table lookup, some redundant nodes can be aggregated or deleted to reduce the size and the depth of tree structure.

As a well-known reduced binary tree, Patricia tree and Trie have been studied [9]–[11]. A Patricia tree, shown in fig.2 (a), can aggregate redundant intermediate node. A Trie, shown in fig. 2 (b), has results of routing table lookup not only in terminal (leaf) nodes but also in intermediate (branch) nodes. Although there are some kinds of binary trees for routing table lookup [10]–[12], the Patricia tree is most used in research of tree-based routing table lookup.

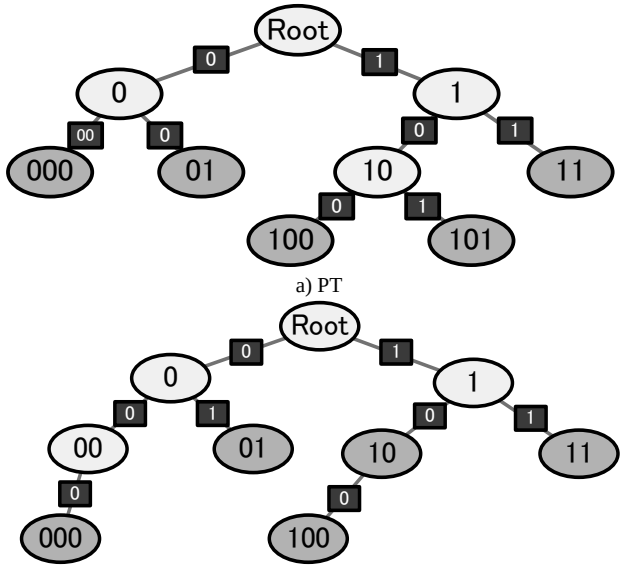
To achieve high-throughput routing table lookup by using a tree structure, enhancement from binary tree structure to a multi-way tree structure [13]–[15] and implementation for pipelined architecture [16]–[18] are known methods of improving lookup throughput. The improvement is mainly achieved by increasing the bit-width of nodes because it can increase the number of pointers per node. By enhancing the binary tree to a multi-way tree, significant reduction of the depth of the tree structure can be expected and the number of redundant memory accesses can be reduced. Adoption of a pipelined architecture is also an effective solution to achieve high-speed routing table lookup. Pipelined access of a tree structure is achieved by assigning a group of sub-trees as a pipeline stage. A reduced binary tree forms the basis of these techniques for high-speed routing table lookup.

D. Enhanced Patricia Tree

Existing trees cannot use empty pointers at an intermediate node efficiently because the empty pointer of children nodes at each node includes the value of the next bit of prefix. This structuring algorithm increases the number of redundant nodes and the depth of the tree structure. Fig.3 shows the case that such an empty pointer is not utilized in existing algorithms. In this case, a node #10 is registered previously. Now, #10 is a target node and has two pointers. Both pointers indicate empty pointers. Then, assume that new entries #1010 is added under the target node #10. If the entry #1010 is added on Trie, a redundant node #101 must be registered because of a constraint that the next prefix must be "0" or "1". On the other hands, a node #101 is not registered on PT and

EPT because of any prefix-length constraint.

Then, assume that the node #10101 is added on Trie and PT after being added the node #1010. On Trie and PT, although one of the pointers is empty, the entry cannot be registered on the empty pointer because the node #10101 must be registered under the more matched node #1010. However, this structuring algorithm increases the depth of tree structure.



a) PT

(b) Trie

Fig. 2. Example of PT and trie

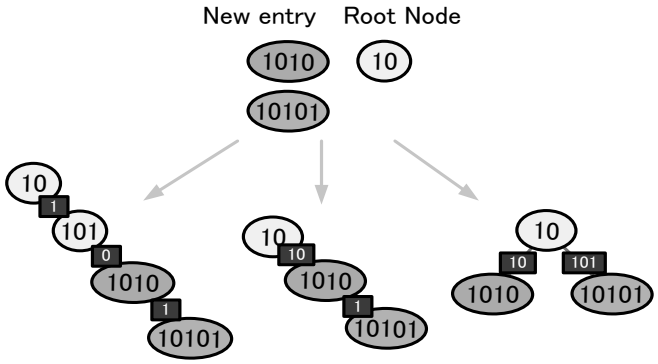


Fig. 3. Example of showing the priority of EPT

To resolve this problem, Enhanced Patricia Tree (EPT) has been proposed [1]. The following is a concept of EPT.

If any pointer is empty, a new entry is registered on the empty pointer possible.

This registration does not matter whether the next bit of the prefix is "1" or "0".

The merit of the former feature is shown in Fig. 3. Node #10101 is registered under node #10 directly. This structuring method can utilize empty pointers and reduce the depth of the tree. However, this registration method may fill a pointer by registering an entry which originally should not be registered to the pointer. For example, in the case of Fig. 3, node #10 has no pointers for node #101 and #100. In order to resolve this problem, EPT must be reconstructed.

In our previous study, EPT is structured and evaluated by using a real BGP table. As a result, compared with existing trees, EPT can shift some nodes to an upper layer as shown in Fig4. Since this shift indicates that the result of routing table lookup can be found in earlier step of memory accesses, EPT can improve Trie and PT. Indeed, compared with PT, the count of memory accesses of routing table lookup is reduced.

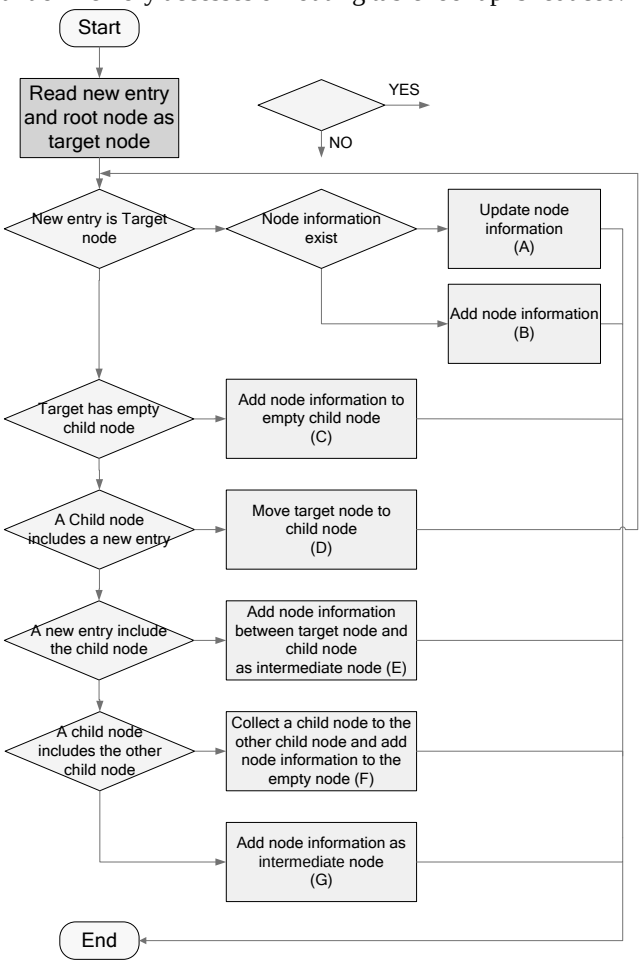


Fig. 4. Flow chart of incremental update algorithm

III. Incremental Update Algorithm

Incremental and decremental update algorithms take a different approach. Incremental update algorithm is an algorithm to insert an entry. In this section, the incremental update algorithm is proposed. Fig. 4 and 5 shows the procedure of the incremental update algorithm. The details of proposed structuring algorithm are as follows.

1) New entry for incremental update is fetched. Root node is read as target node. This target node is compared with a new entry and the target node is shifted recursively.
2) If new entry is equal to a target node and node information exists, node information is updated (Case A in Fig. 5).
3) If new entry is equal to a target node and node information does not exist, node information is added (Case B).
4) If the target has empty child node, node information is added to empty child node (Case C).
5) If a child node includes the new entry, the target node is moved to the child node (Case D) and the incremental process is recursively executed.
6) If the new entry includes a child node, node information is added between the target node and the child node as intermediate node. After this addition, the generated empty node is filled with the nodes on the other child

node according to the tree constructing algorithm of EPT-R (Case E).

7) If a child node includes the other child node, the including child node is collected to the other child node and an empty node is made. The node information is added to the empty node (Case F).
8) If both the target and a child node include new entry node, node information is added as intermediate node (Case G).

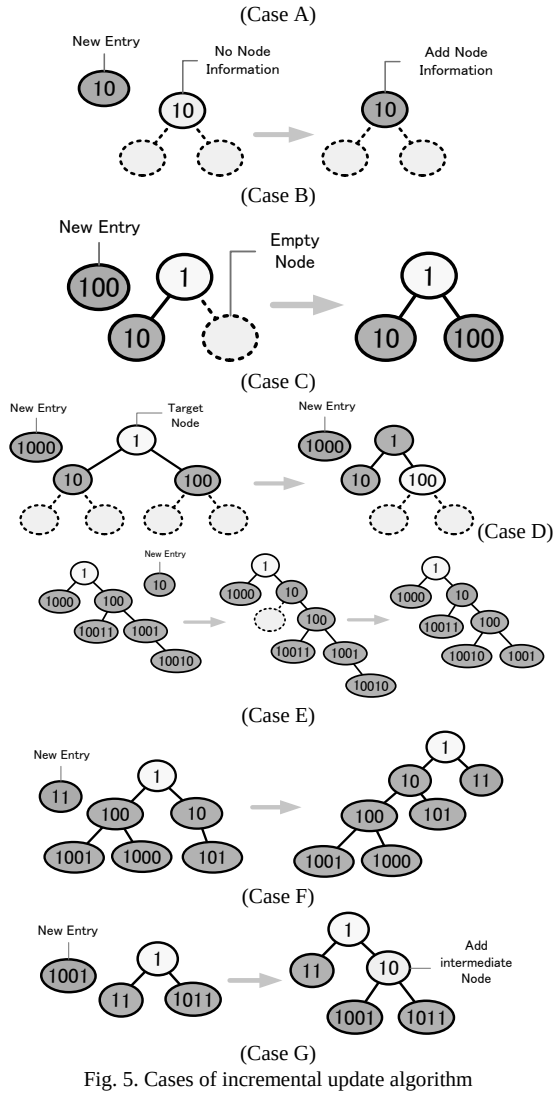


Fig. 5. Cases of incremental update algorithm

IV. Decremental Update Algorithm

In this section, decremental update algorithm is proposed. Fig. 6 and 7 shows the procedure of proposed structuring algorithm. The details of proposed structuring algorithm are as follows.

1) New entry for decremental update is fetched. Root node is read as a target node.
2) If the entry is equal to a child node and both grandchild nodes exist, child node information is updated (Case A in Fig. 7) and go to 4.
3) If the entry is equal to the child node and a grandchild node exists, the child node is replaced with a grandchild node (Case B) and go to 4.
4) If the entry is equal to the child node and no grandchild node exists, the child node is replaced with node on a grandchild node (Case C) which has a longer prefix length and go to 4.
5) If a child node includes the new entry, target node is moved to the child node (Case D) and the decremental process is recursively executed.
6) If both child nodes do not exist, a target node must be merged and go to 5.
7) If target node is not a root node, target node is moved to parent node. And, if target node is root node, the decremental update is finished.

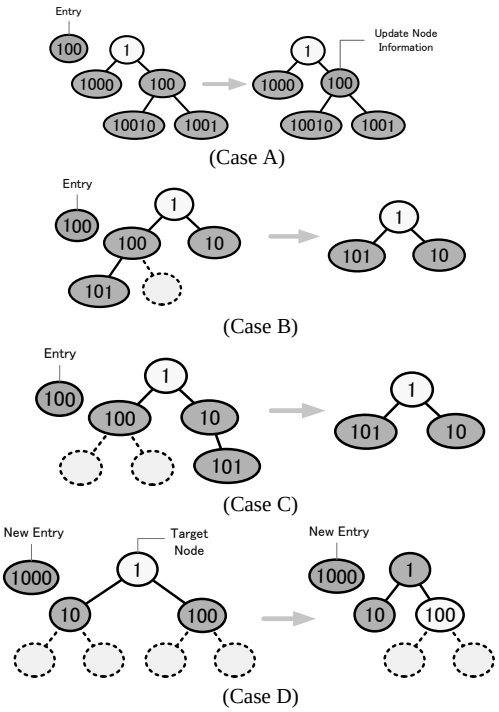


Fig. 7. Cases of decremental update algorithm

V. Evaluation of Update Algorithm

In this section, these proposed algorithms are evaluated for confirming the effect of incremental and decremental update by using real BGP update list. As a compulsion of proposed tree updating method, the existing method reconstructs its tree structure in every updates because the existing algorithm does not support incremental and decremental updates. From this limitation, both normal BGP table and BGP update list are merged to make a complete BGP table for achieving the existing method.

The list of AS1221 table {2010, 07-Jul 16:10:08 (UTC +10:00)} is used as an initial list. This list includes 328762 entries related to ASN-TELSTRA Telstra Pty Ltd. The list of AS1221 update log {2010, 29-Jun 02:00:47 ~ 06-Jul 19:16:39 (UTC+10:00)} is used as update list. This list includes 645 entries for AS1221 incremental update and 85 entries for AS1221 decremental update. The trace log {2007/01/09 7:45~8:15, WIDE} is used as a router access log for evaluating memory access. Intel(R) Xeon(R) CPU E5430 @ 2.66GHz x 4 and jdk1.6.0_20 are used in the evaluation of both proposed method and existing method.

TABLE I: EVALUATION OF INCREMENTAL AND DETRIMENTAL UPDATE ALGORITHM

	Construct Time (us/table)	Decremental Update Time (us/entry)	Incremental Update Time(us/entry)	Memory Access (time)
EPT	8.3899×10^{4}	3.6482	4.5767×10^{-1}	1.7383 x 10
EPT-R	2.2245×10^{5}	2.6964	6.4744×10^{-1}	1.1406 x 10

TABLE I shows the update time of the existing and the proposed method. The average update time of the existing method was 8.3899 x 104 us/table in EPT. Its incremental and decremental update time was 4.5767 x 10-1 us/entry and 3.6482 us/entry, respectively. The average update time of the existing method was 2.2245 x 105 us/table in EPT-R. Its incremental and decremental update time of the proposed algorithm was 6.4744 x 10-1 us/entry and 2.6964 us/entry, respectively. The incremental and decremental update time of the proposed method in EPT was about 23,000 and 183,000 times faster than that of existing method, respectively. The incremental and decremental update time of the proposed method in EPT-R was about 82,000 and 346,000 times faster than that of existing method, respectively. Therefore, the proposed incremental and decremental update algorithms for EPT and EPT-R are effective from the view point of the update time

VI. CONCLUSION

In this paper, a routing table managing algorithm was proposed. The correctness of algorithms was confirmed by using a real trace of internet backbone traffic and real BGP table list. The average update time of the proposed method was at least 23,000 times faster than that of existing method in EPT and EPT-R. These proposed incremental and decremental update algorithms for EPT and EPT-R are effective from the view point of the update time. And, these faster updates could solve a problem in the future real-time Internet service, especially for achieving a quick handover or preventing security attacks, such as anti-phishing function.

REFERENCES

[1] D. Akashi and H. Nishi, "Routing lookup by using enhanced patricia tree," in *Proc. of the Third International Conference on Future Internet (CFI2008)*, June, pp. 105-106, 2008.

[2] D. Akashi and H. Nishi, "Improvement of structuring algorithm on enhanced patricia tree," in *Proc. of the Third International Conference on the Latest Advances in Networks*, December 10-12, pp. 147-152, 2008.

[3] K. Pagiamtzi and A. Sheikholeslami, "Content-Addressable Memory (CAM) circuits and architectures: a tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712-727, 2006.

[4] V. C. Ravikumar, R. N. Mahapatra, and L. N. Bhuyan, "EaseCAM: an energy and storage efficient TCAM-based router architecture for IP lookup," *IEEE Trans. on Computers*, vol. 54. no.5, pp. 521-533, 2005.

[5] "AS65000 BGP routing table analysis report: Active BGP entries (FIB)," [Online]. Available: http://bgp.potaroo.net/as1221/bgp-active.html

[6] I. L. Chvets and M. H. M. Gregor, "Multi-zone caches for accelerating IP routing table lookups," in *Proc. of the workshop on the High Performance Switching and Routing (HPSR)*, pp.121-126, 2002.

[7] B. Talbot, T. Sherwood, and B. Lin, "IP caching for terabit speed routers," in *Proc. of the Global Communications Conference (Globecom'99)*, pp.1565–1569, 1999.

[8] M. Okuno, S. Ishida, and H. Nishi, "Low-power network-packet-processing architecture using process-learning cache for high-end backbone router," *IEICE Trans. on Electronics*, vol. E88-C, no. 4, pp. 536-543, 2005.

[9] M. Kikuta, "Patricia tree," *Journal of Japan Society of Artificial Intelligence*, vol.11, no.2, pp. 337-339. (In Japanese), 1996.

[10] M. A. R. Sbnches, E. W. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, no. 2, pp. 8-23, Mar/Apr 2001.

[11] K. Sklower, "A tree-based packet routing table for Berkeley Unix," in *Proc. of the 1991 Winter USENIX Conference*, pp. 93-99, 1991.

[12] L. Wuu, K. Chen, and T. Liu, "A longest prefix first search tree for IP lookup," in *Proc. of the IEEE International Conference on Communications (ICC2005)*, vol. 2, pp. 989-993, 2005.

[13] W. Lu and S. Sahni, "Recursively partitioned static IP router-tables," *IEICE Trans. on Computers*, vol. 59, no. 12, pp. 1683-1690, 2010.

[14] V. Srinivasan and G. Varghese, "Fast address lookups using controlled prefix expansion," ACM *Trans. on Computer Systems*, vol. 17, no. 1, 1999.

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

[16] K. S. Kim and S. Sahni, "Efficient construction of pipelined multibit-trie router-tables," *IEEE Trans. on Computers*, vol.56, no. 1, pp. 32-43, Jan, 2007.

[17] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proc. of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '98)*, March-April, vol. 3, pp.1240-1247, 1998.

[18] A. Basu and G. Narlikar, "Fast incremental updates for pipeline forwarding engines," in *Proc. of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*

[19] *(INFOCOM2003)*, vol.1, pp. 64-74, 2003.

Shin-ichi Ishida received his B.E., M.E. degrees from Keio University, Japan, in 2005 and 2008, respectively. He is now a Ph.D candidate in the school of integrated design engineering at Keio University. His research interests include the intelligent backbone Internet and router micro architecture.

Koji Ikehara received his B.E., M.E. degrees from Keio University, Japan, in 2009 and 2011, respectively. He is currently a staff in the Hitachi Ltd. The main theme of his current research is in high performance router architecture

Hiroaki Nishi received his B.E., M.E., and Ph.D. degrees from Keio University, Japan, in 1994, 1996, and 1999, respectively. He has been selected as fellowships from Japan Society for Promotion of Science from 1996 to 1999. He was a researcher in Real World Computing Partnership from in 1999 and researched high performance network for cluster computers. Since 2002, he was a researcher in the Central Research Laboratory, Network Platform Department, Hitachi Ltd., and he led new generation backbone router project. He was Lecturer and Assistant Professor in Department of System Design Engineering, Keio University in 2003 and 2006, respectively. Now, he is Associate Professor of Keio University since 2007 and Visiting Associate Professor of National Institute of Informatics (NII), Japan since 2010. Prof. Nishi is a member of IEEE, Institute of Electrical Engineers of Japan, Architectural Institute of Japan, Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers and the Society of Instrument and Control Engineers. He received Network System Research Award in 2003 from IEICE, Award for the outstanding contribution IEEE ICS Advanced Motion Control Workshop in 2004, Best Paper Award of FANAC FA Robot financial group in 1007 and 2008, Best Paper Award of IPSJ Ubiquitous Computing Society in 2010, Best Presentation Award of IPSJ SLDM society in 2010 and Computer System Award of IPSJ in 2011. The main theme of his current research is in building of the total network system including development of hardware and software architecture.