Evaluation and Comparison of Binary Trie base IP Lookup Algorithms with Real Edge Router IP Prefix Dataset

Alireza Shirmarz
Department of Computer Engineering and Information, Amirkabir University of Technology
Tehran, Iran

Masoud Sabaei
Department of Computer Engineering and Information, Amirkabir University of Technology
Tehran, Iran

Mojtaba hosseini
Department of Computer Engineering and Information, Amirkabir University of Technology
Tehran, Iran

***Abstract*—Internet network is comprised of routers that forward packets towards their destinations. IP routing lookup requires computing the Best-Matching Prefix. The main Functionality of Router is finding the Appropriate Path for Packet. There are many Algorithms for IP-Lookup with different Speed, Complexity and Memory usage. In This Paper Three Binary Trie algorithms will be considered for Performance Analysis. These algorithms are Priority-Trie, Disjoint Binary and Binary Trie. We consider three parameters for comparison, these parameters are Time, Memory and Complexity of Algorithms. For performance analysis, we develop and run algorithms with real Lookup-Tables which were used in an edge router.**

***Keywords*—Binary Trie; IP-Lookup; Running Time; Memory Usage; Complexity**

I. Introduction

Routers Receive Packets and Analyze them. Routers Extract Source and Destination IP Address from packets. This equipment looks for Destination IP Address in a Table with a lot of IP Prefixes, Output Interfaces and Next-Hops IP Address, then Forwards Packets to next-Hop through the output interface. Routers fill Lookup Table With many Different Routing Protocols that are Standard. IP Lookup is a bottleneck of Routing [1]. There are two Versions of IP Address, IPv4 & IPv6. In this Paper concentration is on IPv4 that is universal and popular. IPv4 structure consists of 32 bits and is shown with 4 decimal digits that are separated by dot [2]. There is Two Type of IPv4 addressing that are defined in (a) and (b).

A. Classful IP Addressing

IPv4 Structure divided by two portions. More Valuable bits portion is network section and the other one is for Hosts. Network bits are called "Address Prefix". For showing IP Address Prefix, bits related to Network are used and for other bits "*" is used, for example 1000001001010110* is an IP Address Prefix that consist of 2^{16} IP Address that are started by this IP Address Prefix. Address Prefix is shown in Decimal like 130.86/16 [1]. Routers should search and Forward Packets with IP address Prefix.

TABLE I. Forwarding Table Example [1]

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

Some Classes are defined in this addressing structure, for example IP class A, B & C are shown in fig (1).

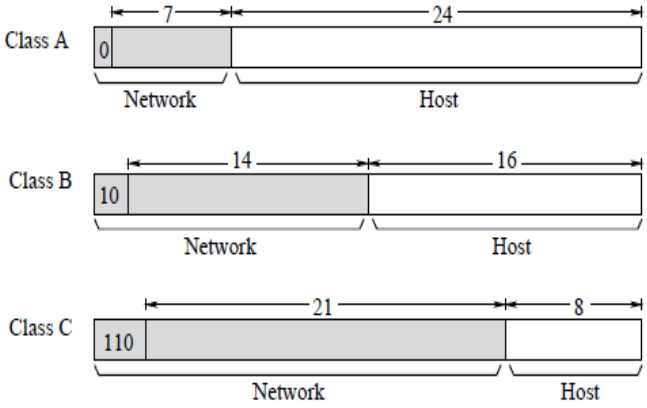

Fig. 1. Classful IP Addressing [1]

B. CIDR IP Addressing

This addressing method is proposed for efficient use of address space in IPv4. In classful addressing, there are limited classes but in CIDR addressing method, network bits are variable and specify with Mask. IP address and Net Mask are operated by XOR and network bits extracted.

Routers must use Longest Prefix Matching for IP Lookup that is shown in fig (2).

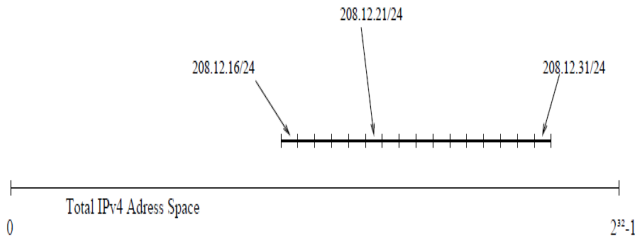

Fig. 2. CIDR Addressing in IPv4 Space [1]

Paper [5] proposed an optimized method for IP Lookup Table management with considering of memory complexity.

In this Paper, three algorithms which are Binary Trie Base are developed and run for twelve times, and evaluate algorithms in a situation with real IP Lookup Table Data. This IP lookup Table data is collected in 4 period of time and data variable is considered. Datasets are used for each algorithm. In section 2, the algorithms are described and in section 3, the situations of development and simulation are described and in section 4, the results and Charts are shown then in the last section there is a conclusion .In the last section there is a conclusion about the results.

Router processes entry IP Packet and extract IP address. It uses IP address prefix or IP network section and looks for it in lookup table then forwards packet to specified output interface. Totally there are two type IP Lookup algorithms, hardware and software base. table (II) shows IP lookup algorithms.

TABLE II. TOW TYPE IP LOOKUP ALGORITHMS (HARDWARE AND SOFTWARE)

Hardware Base Algorithms	Software Base Algorithms
DIR-24-8-BASIC Scheme	Binary-Trie
DIR-Based Scheme with Bitmap Compression (BC-16-16)	Path Compressed Trie
Ternary CAM for Route Lookup	Multi-Bit Trie
Algorithms for Reducing TCAM Entries	Level Compression
Reducing TCAM Power – Cool CAMs	Lulea Algorithm
TCAM-Based Distributed Parallel Lookup	Tree Bitmap Algorithm
	Tree-Based Pipelined Search
	Binary Search on Prefix Range

Nowadays SDN separates Data Plane and control Plane, so software base algorithms got important contrary to advantage of Hardware base algorithms. SDN make service providing agile [3]. NFV try to implement network functions as software and virtual [4]. Software base network simplify network entity. Because of software base network importance, three software base and basic IP Lookup algorithms are developed, Evaluated and compared in this paper.

Three basic and main software base IP Lookup algorithms are Binary trie, Disjoint Binary Trie & Priority Trie that other software base algorithms derived from them.

C. Binary Trie algorithm [2]

First, this algorithm reads IP Address Prefixes from IP Lookup Table and makes a Binary tree then looks for per entry packet IP address. The tree structure that this algorithm makes is shown in fig (3).

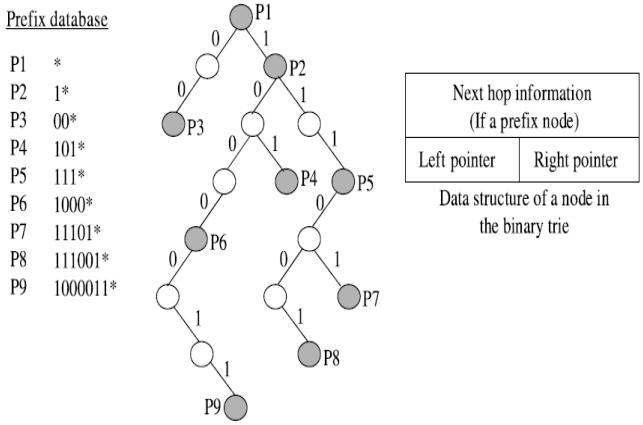

Fig. 3. Binary Trie [2]

- Binary Trie Prefix construction Algorithm

IP prefix binary tree construction needs a structure with three fields.

1) *A Pointer to the left*
2) *A pointer to the right*
3) *A value with binary type*

Preprocessing of IP lookup table to make sure all prefixes are in binary format. To begin, a software module reads each prefix from IP lookup file and a pointer creates as a root node then if current bit is 0 pointer goes to the left branch else if current bit is 1 pointer goes to the right branch. Pointer reads next bit and base on value does as mentioned algorithm before from the current node. In each IP address prefix last bit, pointer writes the prefix in value variable.

- IP Lookup algorithm base on binary tree

In first step, a pointer reads entry IP address valuable binary and if pointer sees 0 then points to left branch else if pointer sees 1 then points right branch. This algorithm goes on until pointer points a node that doesn't exist. if the pointer sees a node with value (not null) this prefix is longest prefix match for this entry IP address else if pointer sees last node with null value, it should register last node value as longest prefix matching that has seen before and has not been null.

- Insert node to the tree

For insertion also navigate the tree like construction and lookup algorithms and add value null for intermediate node and insert specified value for the last node of tree.

- Delete node from the tree

For this algorithm like making, lookup and insertion, tree should be navigated and deleted node and delete intermediate node until a node with not null value is seen.

Memory complexity depends on number of address prefix in lookup table and number of prefix bits O(NW). N is number of prefixes in lookup table and W is shown number of prefix bits. Memory complexity depends on memory speed and number of memory access. Algorithm Complexity depends on number of instructions that are run.

D. Disjoint Binary Trie [2]

This algorithm is similar to Binary Trie with a difference that there is a full binary tree and all leaves are prefix except intermediate node which is shown in fig (4).

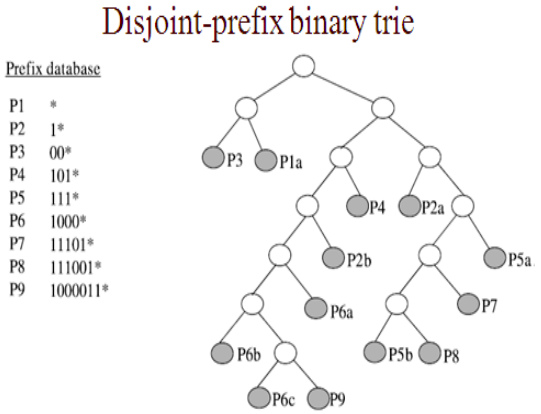

Fig. 4. Disjoint Binary Tree [2]

In this method pre-order tree navigation is done and if intermediate node has a not null value, registers and insert s to subtree. This algorithm use 128 node for tree and memory complexity increases but tree gets more structural.

E. Priority Trie Algorithm [6]

This algorithm like Binary Trie does but there is a difference in tree node arrangement that is done as descending mode in tree. Nodes with null value are deleted. In the binary tree, there is two problems, first existence of many nodes with null value and second Long and deep prefix with too delay in lookup, all these problems are solved in this algorithm.

First prefixes should sorted descending mode with considering of prefix length. A tree is made like binary tree. In this algorithm, intermediate node is not needed and where a prefix is set to each node until decrease tree navigation, fig(5) shows this concept.

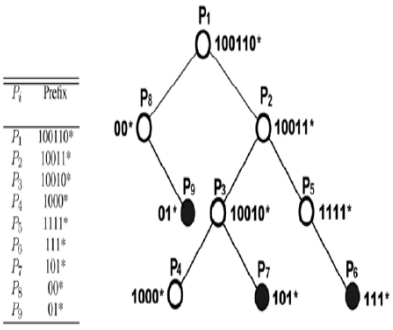

Fig. 5. Priority Trie [6]

In this algorithm a node with more prefix length in comparison of tree level, is called Priority node and seen as a white circle in fig(4) and a node with less prefix length in comparison of tree level, is called Ordinary and is shown with black circle in fig(5).

II. MATERIALS AND METHODS

In this paper, C++ language is used for IP Lookup algorithm Implementation. All IP Lookup table data are collected from an edge router in four different months. IP Prefixes have saved in a text file.

In this project a node is implemented by a class. This class consists of two pointers and a variable called value. Pointer is node class type and Value variable is array of 32 Boolean type variables.

An instance of class is defined to use in node construction for binary tree. In priority algorithm, a bit is defined to specify node type.

For calculating of memory complexity, number of nodes multiply used memory of per node therefore memory complexity of Binary tree is equal:

$$storage\ Coplx(bit)=(32*2+Value.length*16);$$

Each Node Complexity (1)

In Priority Trie algorithm 1 is added to the above equality. in 32 bit OS, a pointer occupies 32 bit and in 64 bit OS, a pointer occupies 64 bit that should be considered in memory complexity calculation. We use a 32 bit OS.

For calculating time, we run software and calculate spent time.

For calculating of algorithm complexity, we use the number of instruction for each IP lookup algorithm.

III. RESULTS

We have four IP Lookup Prefixes in text file and we read and make tree for different IP lookup algorithms.

A. 1th IP Lookup table

This dataset has 519998 prefixes.

TABLE III. FIRST THREE ALGORITHMS COMPARISON

	Time	Complexity	Memory(Hop*Mem)	Total Memory(Bit)
Binary	12140.63	59198146	312768	206786368
Disjoint	10250	60916464	403591	201097312
Priority	38812.5	69625952	204613	16521520

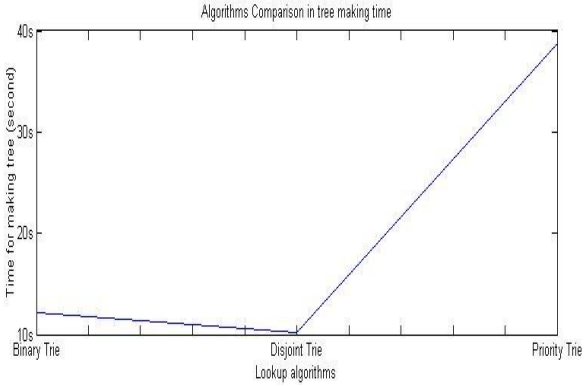

Fig. 6. Comparison of three algorithms In Tree making time (ms)

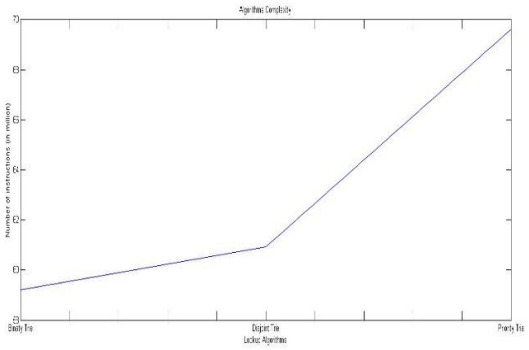

Fig. 7. Comparison of three algorithms Complexity

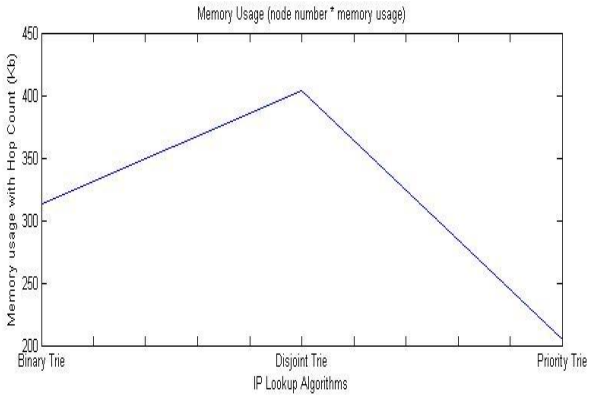

Fig. 8. Comparison of three algorithms In memory complexity

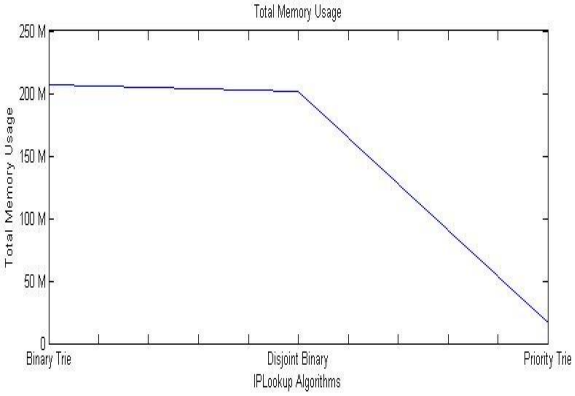

Fig. 9. Comparison of three algorithms In memory usage (bit)

B. 2th IP Lookup table

This dataset has 520000 prefixes.

TABLE IV. SECOND THREE ALGORITHMS COMPARISON

	Time	Complexity	Memory(Hop*mem)	Total Memory(Bit)
Binary	11875	57472334	314766	207073184
Disjoint	10109.38	59199886	407819	206869728
Priority	38750	67329984	206206	16588736

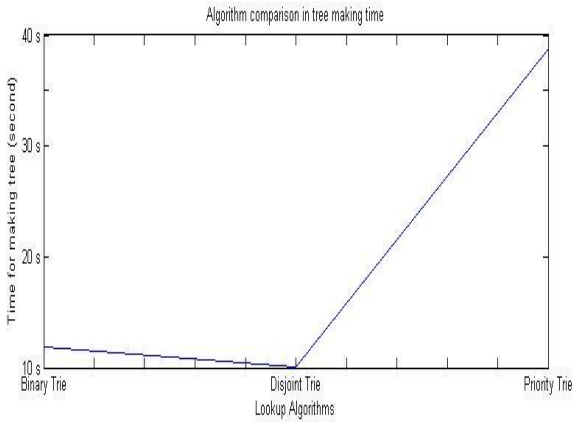

Fig. 10. Comparison of three algorithms In Tree making time (MS)

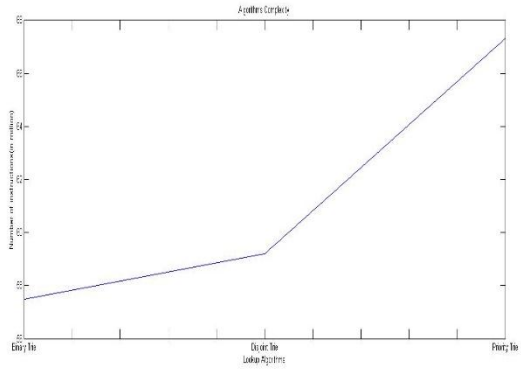

Fig. 11. Comparison of three algorithms Complexity

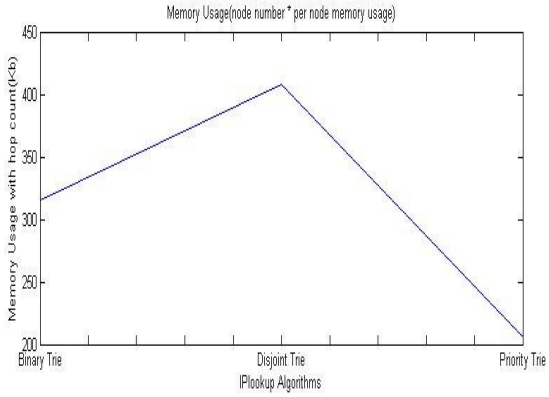

Fig. 12. Comparison of three algorithms In memory complexity

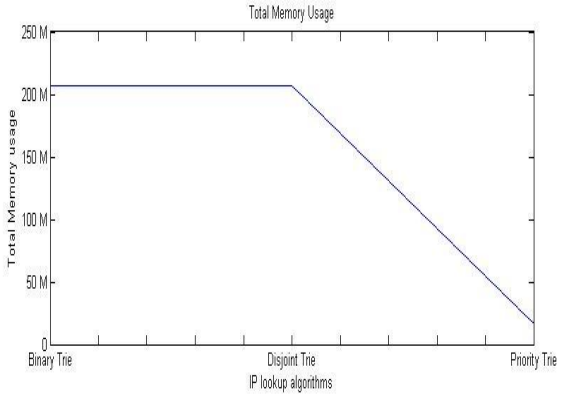

Fig. 13. Comparison of three algorithms In memory usage (bit)

C. *3th IP Lookup table*

This dataset has 519995 prefixes.

TABLE V. THIRD THREE ALGORITHMS COMPARISON

	Time	Complexity	Memory(Hop*Mem)	Total Memory(Bit)
Binary	11296.88	58611469	327324	205794016
Disjoint	9750	60407101	429595	204547792
Priority	38046.88	68800571	208218	16820048

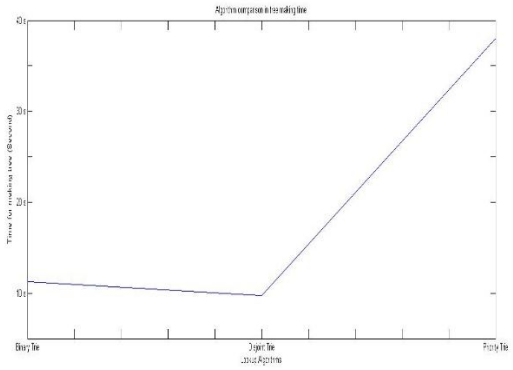

Fig. 14. Comparison of three algorithms In Tree making time (MS)

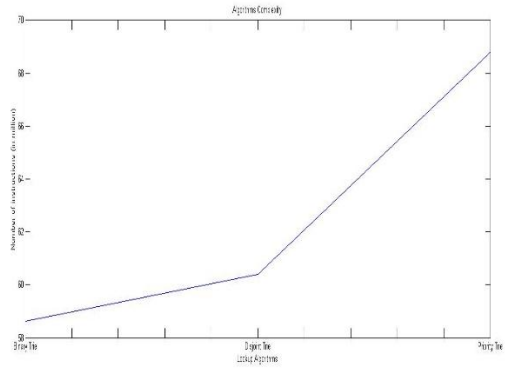

Fig. 15. Comparison of three algorithms Complexity

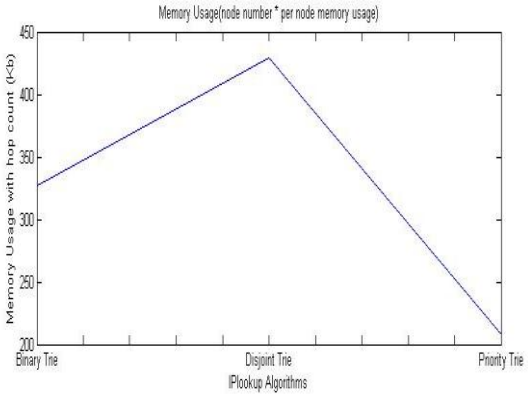

Fig. 16. Comparison of three algorithms In memory complexity

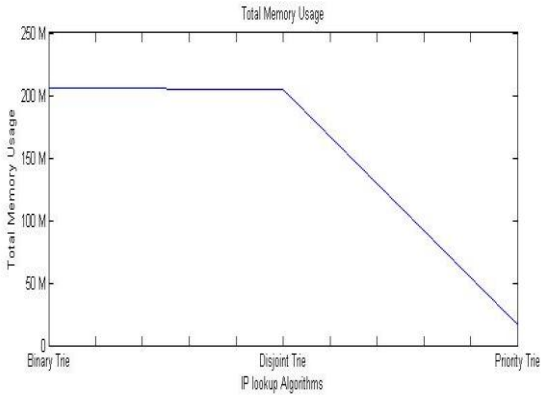

Fig. 17. Comparison of three algorithms In memory usage (bit**)**

D. 4th IP Lookup table

This dataset has 519998 prefixes.

TABLE VI. FOURTH THREE ALGORITHMS COMPARISON

	Time	Complexity	Memory(Hop*mem)	Total Memory (Bit)
Binary	12140.63	59198146	312768	206786368
Disjoint	10250	60916464	403591	201097312
Priority	38812.5	69625952	204613	16521520

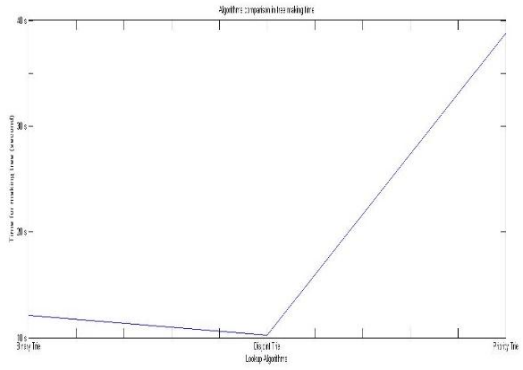

Fig. 18. Comparison of three algorithms In Tree making time (MS)

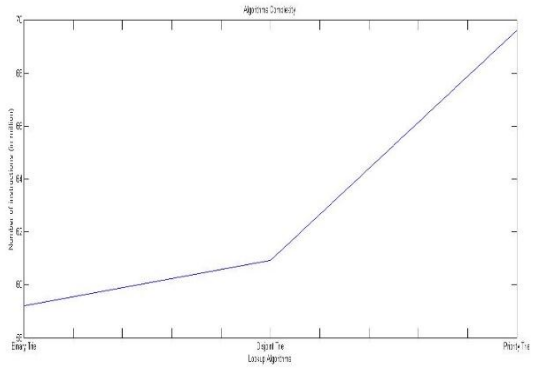

Fig. 19. Comparison of three algorithms Complexity

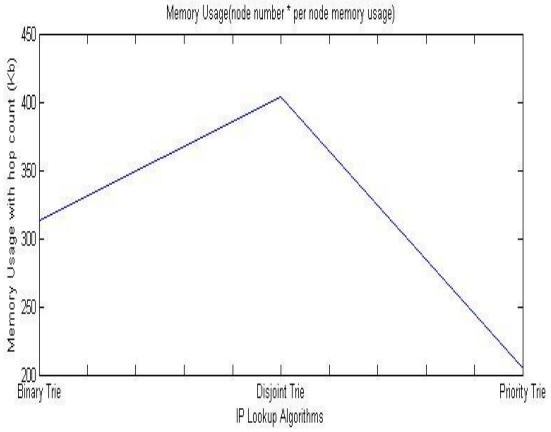

Fig. 20. Comparison of three algorithms In memory complexity

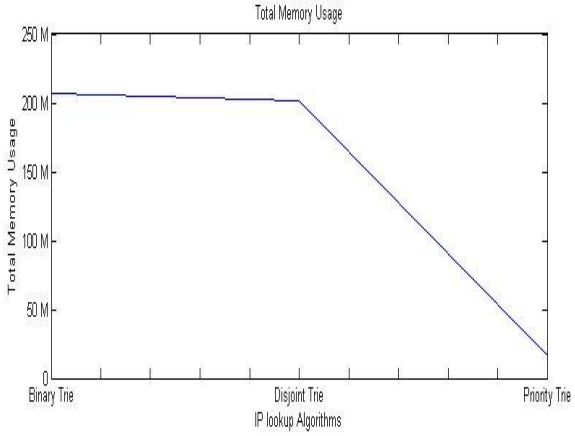

Fig. 21. Comparison of three algorithms In memory usage (bit)

IV. DISCUSSION

According to the results, Internet network is comprised of routers, that forward packets towards their destinations. IP routing lookup requires computing the Best-Matching Prefix. The results show that the Priority Trie algorithm performance is need more time to make tree in compared with other two algorithms because of sorting. Binary trie and Binary disjoint trie algorithms need similar time to make tree but if tree goes

toward full binary tree, more similarity was seen and if tree goes toward light, more different time was seen.

From the point of algorithm complexity, Priority trie algorithm based on number of instruction is the most complex algorithm in comparison with the others.

Disjoint binary trie algorithm has the most memory usage because there are more leaves in comparison with the others. Priority algorithm has the least memory usage in comparison because deleted leaves value is null.

Priority trie IP lookup speed is more than the other algorithms. When there is longer prefix, Priority trie algorithm search speed is much better than the others.

For the future work, two subjects are proposed:

A. Evaluation of binary trie base algorithms with use of IPv6,

B. Evaluate the other algorithms that are described in this paper for IPv4.

REFERENCES

[1] 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, Mar/Apr 2001.

[2] doi: 10.1109/65.912716.

[3] H.Jonathan Chao, Bin Liu, “High Performance Switch And Routers,” in Book 2007.

[4] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: a survey", Communications Surveys & Tutorials, vol. 16, no. 1, pp. 493-512, 2014.

[5] M. K. Shin, Y. Choi, H. H. Kwak, S. Pack, M. Kang and J. Y. Choi, "Verification for NFV-enabled network services," Information and Communication Technology Convergence (ICTC), 2015 International Conference on, Jeju, 2015, pp. 810-815.

[6] doi: 10.1109/ICTC.2015.7354672.

[7] Kun Huang, Gaogang Xie, Yanbiao Li, Dafang Zhang, “Memory-efficient IP lookup using trie merging for scalable virtual routers” in Journal of Network and Computer Applications, vol. 51, pp.47-58, May 2015.

[8] H. Lim ; Ewha Womans University, Seoul ; C. Yim ; E. E. Swartzlander Jr, “Priority Tries for IP Address Lookup” in IEEE Transactions on Computers, vol. 59, no.6, February 2010.