Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2016, 3(7): 28-33

Review Article ISSN: 2394 - 658X

Review on Tries for IPv6 Lookups

Rohit G Bal

Department of Computer Science, Nepal Engineering College, Nepal
rohitgb@nec.edu.np

ABSTRACT

Main task of a Router is to provide routing of Internet Protocol (IP) packets. Routing is achieved with help of the IP lookup. Router stores information about the networks and interfaces in data structures commonly called as routing tables. Comparison of IP from incoming packet with the IPs stored in routing table for the information about route is IP Lookup. IP lookup performs by longest IP prefix matching. The performance of the IP router is based on the speed of prefix matching. IP lookup is a major bottle neck in the performance of Router. Various algorithms and data structures are available for IP lookup. This paper is about reviewing various tree based structure and its performance evaluation.

Key words: IP-address lookup, tries, IP packet classification, performance, IP networks

---

INTRODUCTION

Routers role is to forward the Internet Protocol (IP) packets to the destination based on the IP header of ingress packets coming in to the router. In router there is a routing table which stores information about the interfaces and IP addresses. Router should decide where the packet should send, for this router should compare the entries of routing table with the destination IP address from IP header of ingress packets. This basically is searching of the routing table with key as destination IP address. The above mentioned operation is termed as IP Lookup. Router will forward the IP packets to interface mentioned after forwarding interface information is retrieved routing table.
Internet is doubling every two years by Moore's law of data traffic [1] and the number of hosts is tripling every two years [2]. Threats are also growing proportionally over the years. More and more data are stored digitally nowadays. In Wide Area Network (Internet) is numerous remote host access system with help of network. The rapid growth of internet leads to difficulty in IP lookup in router. Routing is stressed because of the rate of incoming packets is exponentially increasing yearly. The improvement in the physical media is also case of better data rates which leads to improve the router performance. The routers performance can be improved by optimizing IP lookup. For improving the IP lookup scheme we consider two options mainly.

- Optimizing Data Structure
- Optimizing Search Algorithm

This paper mainly focus on the reviewing various data structure proposed for IP lookups in the router. Main discussions of this paper are Binary Tire, Path Compression Tire, Prefix Expansion Tire, Disjoint Prefix Tire and Prefix Hash Tire.

IPv6

IP version 6 (IPv6) [3] is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4) [4]. IPv6 solves the address depletion problem of IPv4. In IPv4 maximum number of IP that can be used is up to 232 while in IPv6 it is up to 2128. IPv6 address has length of 128 bit which is divided into 8 parts. Each part is 16 bit long. Each part is represented using hexadecimal value where minimum value of each part is 0000 and maximum value is FFFF. With IPv6 6.65 x 10^{23} addresses are available for every square meter of the Earth's surface.

IPv6 Header

IPv6 header is of 320 bit in length. It consist of information such as version, traffic class, flow label, payload length, and next header, hop limit, source address and destination address [5].
Version (4 bit): Indicates version (6) of IP packet.

Traffic Class (8 bit): Facilitates the handling of real time data by router. IT Prioritize the packets.
Flow Control (20 bit): Used to label sequences of packets that require the same treatment for more efficient processing on routers.
Payload Length (16 bit): Length of data carried after IPv6 header.
Next Header (8 bit): Identifies the higher level protocol
Hop Limit (8 bit): This field indicates how long packet can remain in network.
Source Address (128 bit): This Field indicates the IPv6 address from which packet is generated.
Destination Address (128 bit): This field indicates the IPv6 address to which packet is going.

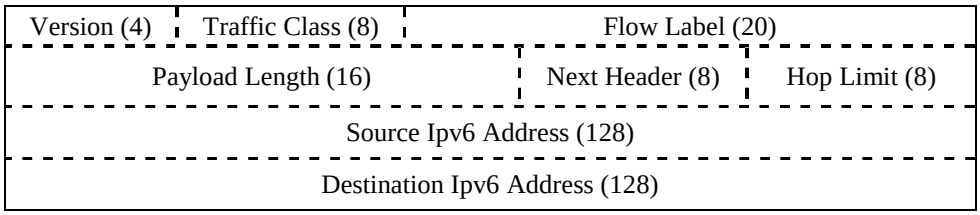

Fig. 1 IPv6 Packets Header Format

Address Prefix: The Address prefix is similar CIDR value in the IPv4. It is used to represent express routes, address ranges and address spaces.

Ipv6-address/prefix-length

Where is, **ipv6-adress** is 128 bit address **prefix-length** values varies from 1 to 128

Address Types: The type of an IPv6 address is identified by the high-order bits of the address, as follows:

Loopback -- ::1/128
Multicast -- FF00::/8
Link-local unicast -- FE80::/10
Site-local unicast -- FEC0::/10
Global unicast -- All other
Global Unicast Address

We are considering mainly global unicast address because most of the routing packets is based on Global unicast address, site-local and link local is more like packets inside a network only

Fig. 2 IPv6 Global Unicast Address Format

Here for the IPv6 address the prefix length is M+N if the M+N bits are similar for 2 IPv6 address those 2 nodes are in same network which needs no routing. And if M+N is different routing is necessary.

IPV6 ROUTING

IPv6 Forwarding Process

The router checks of prefix bit of destination IP of ingress packets IP header. The prefix is compared with the routing table entry. The IP packet is forwarded to the longest matching prefix address's interface.

Routing Table

Router maintains a routing table also called as forwarding table. Routing table is a data structure which stores information of IPv6 address destination [5]. The routing table consist of IPv6 prefix, prefix length, next hop address, next hop interface, metric, timer and route source. IPv6 prefix and prefix length helps to match the route from routing table. Next hop address is the address of other router that is connected directly. Next hop interface is the interface connected to other routers. Metric is an integer number indicating total distance to destination. Timer indicates the last update of routing table. Route source indicates the protocol of routing used.

Router IP Lookup

Router consists of mainly, Incoming interfaces, outgoing interfaces, some functional units, SDRAM and Network processor. Incoming and outgoing interfaces is where packets come and go in a router. Functional units consist of Flash memory for routers OS, Boot ROM for storing booting sequence, and NVRAM for storing configuration. SDRAM stores runtime executable router OS, routing tables, caches and IP packets [9] [11]. Network processor processes the IP packets and IP lookup process is also done here.

Main use of network processor is IP packet classification & IP lookup and routing table lookup. IP packet classification means classify packets according to their header information. For packet classification data structures are used to store information of IP prefix. In backbone routers IP lookup now a day is based on best effort lookup. Best effort lookup in the sense it will not do perfect matching, instead best matching prefix is selected as interface as next hop. IP lookup uses data structures for storing and comparison. Here we are discussing about various data structures used in IP lookup.

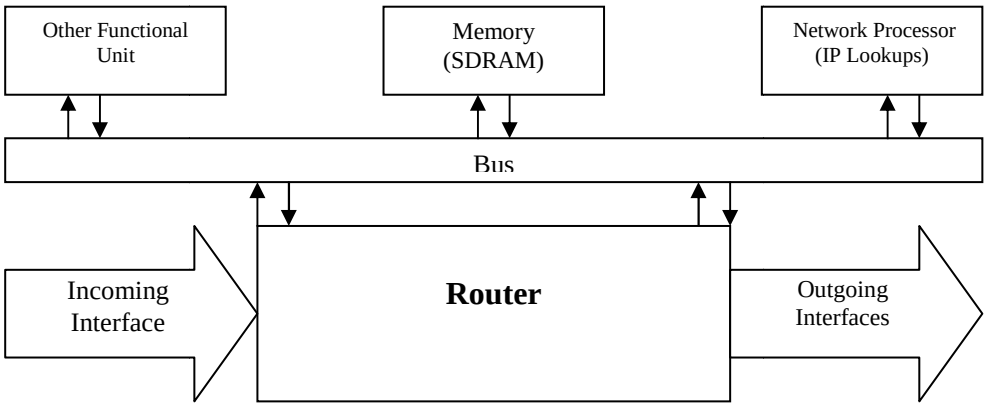

Figure 3: Router Architecture

DATA STRUCTURE AND ALGORITHMS

Data structures are used to store IP addresses in organized fashion inside memory. The Data structures are of 2 types linear and nonlinear. Linear data structure is organized in sequential manner in memory; here one IP address is associated with one IP address. Nonlinear IP addresses are not arranged in sequential order [6]. Linear data structure is not considered for IP lookup in backbone router since it has poor scaling, Time complexity for classifying packet grows linearly with number of IP address stored. In this section we mainly discuss about various hierarchical data structure used in IP lookups.

Requirements for Algorithms

The following are the observations made as requirements for the selection of routing lookup algorithm in networks.

- Storage Efficiency for the IP prefixes
- Should able to store variable length prefixes
- Dynamic insertion and deletion for dynamic routing table updates.
- Worst case retrieval should be independent of prefix size.

Binary Tries

Binary tries is an ordered 2 way tree data structure to store strings of bits using either 1/0. In routers IP address is represented in form of strings of bits. Searching in binary tries is done by comparing one bit at a time [8]. IP lookup is basically comparison of bits here best prefix matching. The binary tree will help us to save the data is less amount of space. Binary tree have 3 types of nodes root node, intermediate node and leaf node. Root node represents default routing. Leaf node holds the routing information i.e. the egress interface information. Intermediate nodes are nodes that used for traverse from root node to leaf node. Some leaf node act as intermediate node for other leaf node. The height of the tree depends on the maximum number of bits in IP prefix in the routing table.

Consider packets with destination IPv6 address prefix 01101 comes to router. The lookup process is like this initially the here there is no exact matching is not applicable. There are 2 matching leaf nodes. i.e. P1, and P3 but according to the longest prefix is P3 is selected.

Table -1 IP Prefixes stored in Router

Prefix	Prefix Strings
P1	0*
P2	01000*
P3	011*
P4	1*
P5	100*
P6	1100*
P7	1101*
P8	1110*
P9	1111*

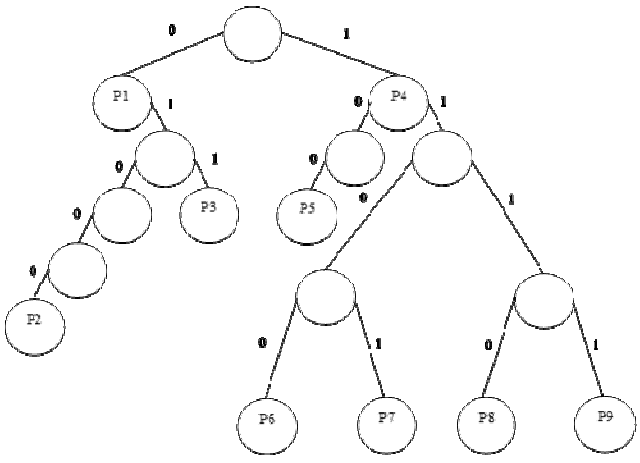

Fig. 4 Binary Tries for the prefixes in Table -1

Path Compressed Tire

Path compression reduces the height of a sparse binary tries. Path compressed tries are binary tries which compress the long sequence of one child node present. It is based on Practical Algorithm to Retrieve Information Coded In Alphanumeric (PATRICIA) [7].Path compression is possible only for sparsely populated trees. When the tree is full and there is no compression possible, a path-compressed tries looks the same as a regular binary tries. Path compression will eliminates the long sequence of child nodes; this will reduce the memory usage. Skip value is used to keep track of bit that should be compared next.

In above example P2, P3 is reduced its height, some of the child nodes are removed. Skip value is used in each node to identify which bit has to be checked after this. Consider prefix 01100 first it will check 0 will reach node P1 then skip 3 bits and check the 3th bit i.e. 1 so P3 is selected. Now perform exact match if there is exact matching then the P3 is selected and packet is forwarded to the interface stored in P3. There should be bit by bit matching and exact matching also need to perform for identify prefix match. Since some part of tries is skipped we need to consider bit wise matching according to skip value and we need to exact prefix match.

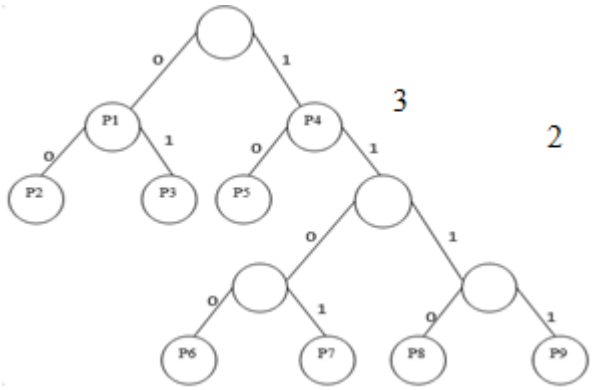

Fig. 5: Path Compressed Tries for the prefixes in Table 1

Prefix Expanded Tries

If the prefixes in the table is of different length the prefixes is expanded to one or more selected common length to avoid different length in lookup. The main rule to follow is not to decrease the prefix length. In Table 1 we have prefixes P1, P4 length 1, Prefixes P3, P5 is of length 3, Prefixes P6, P7, P8, P9 are of length 4, prefix P2 of length 5. Here 4 different prefixes lengths are available (1, 2, 3, 4, and 5). Convert prefix length into some common prefix length (2, 4, and 5). Now prefixes 1 and 2 are converted to length, prefixes 0f length 3 and 4 is converted 4, prefix length 5 is kept intact.

Table -2: Expanded prefixes for prefixes in Table 1

Prefix	Prefix Strings	Remarks
P1	01*	*Converted from length 1 to 2*
P1	00*	*Converted from length 1 to 2*
P2	01000*	*No Conversion in prefix length*
P3	0110*	*Converted from length 3 to 4*
P3	0111*	*Converted from length 3 to 4*
P4	10*	*Converted from length 1 to 2*
P4	11*	*Converted from length 1 to 2*
P5	1000*	*Converted from length 3 to 4*
P5	1001*	*Converted from length 3 to 4*
P6	1100*	*No Conversion in prefix length*
P7	1101*	*No Conversion in prefix length*
P8	1110*	*No Conversion in prefix length*
P9	1111*	*No Conversion in prefix length*

While expanding the prefix all the available combination should be included in the prefix table. So the chance of more usage of memory is required. The routing table entries will be expanded which results in fast searching because of less no of back tracking. These new prefixes have the same interface information as that of the original prefix. Such redundant information can be compressed, saving memory and at the same time making the search faster because of the smaller height of the tries. After compression, the entire data structure can even fit into an L1 cache, which further speeds up the search as the access times.

Fixed Stride Multibit Tries

Fixed stride multi bit tries is variation of path compressed tries but no of bit compressed is fixed in size [12] [13]. Each node in tire have 2s entries where s is no strides. Consider 3 bit stride each node have 8 entries. Consider prefixes in table 1 P1 is 0* so when u expand it into 3 bit it there are 4 possible entries i.e. 000,001,010,011. In the 4 possible entry 011 is already defined as P3 so 011 is replaced by P3 instead of P1 which is expanded. Like that 100 P1 prefix is replace by P5. 110 points to node 2 where p6, p7 prefixes are stored and 111 points to node 3 where p8, p9 prefixes are stored.010 points to node 4 where p2 is stored. Since the prefixes are expanded the multi stride provide better performance by reducing number of memory accesses. It uses more memory since each node have multiple expanded prefixes.

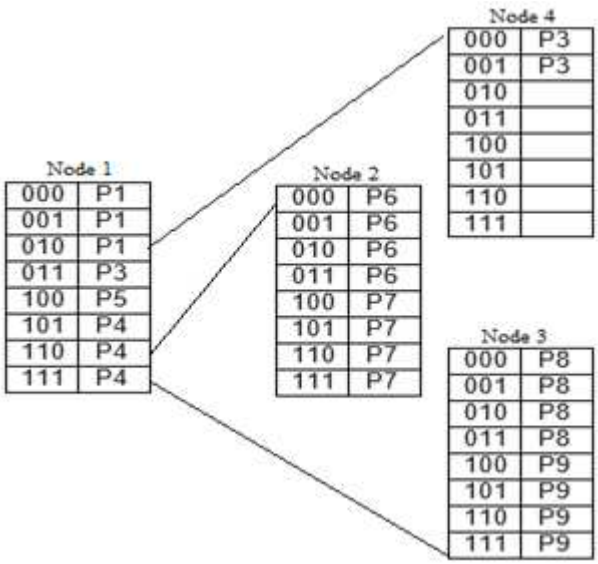

Fig. 6 Fixed stride multibit tries (Stride=3) for the prefixes in Table 1

Variable Stride Multibit Tries

Variable stride multi bit tries is variation of path compressed tries but no of bit compressed is variable in size. Each node in tire has different stride entries. Some nodes have considered 3 bit stride some have 2 some have 1. Total no of entry in a node is 2s where s is the no of strides [10] [12]. Consider prefixes in table 1 P1 is 0* so when u expand it into 3 bit it there are 4 possible entries i.e. 000,001,010,011. In the 4 possible entries 011 is already defined as P3 so 011 is replaced by P3 instead of P1 which is expanded. Like that 100 P1 prefix is replace by P5. 110 points to node 2 where stride is one bit 2 possible entry 0, 1 where p6, p7 prefixes are stored and 111 points to node 3 where stride is one bit 2 possible entry 0, 1 where p8, p9 prefixes are stored and 010 points to node 4 which is 2 bit 4 entries 00, 01, 10, 11 from that 00 is the entry in which p2 prefix is stored. Since the prefixes are expanded the multi stride provide better performance by reducing number of memory accesses. It uses less memory compared since each node has multiple expanded prefixes.

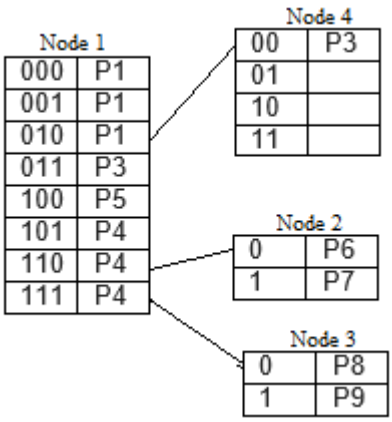

Fig. 7 Variable stride multibit tries for the prefixes in Table 1

CONCLUSIONS

In near future IPv6 will be implemented in Internet and it consist of 128 bit which is lengthier and need more comparison for packet classification. There is a need for more research in field of packet classification and IP lookups for improving the performance of the routers in IPv6 networks. This study will help in research regarding the packet classification in router. In IPv6 networks 128 bit address can be stored in variable stride multibit tries looks a feasible solution but to select the strides for each node should be selected. More research should be applied for stride section automatically.

REFERENCES

[1] KG Coffman and AM Odlyzko, *Internet Growth: Is there a "Moore's Law" for Data Traffic*? Handbook of Massive Data Sets, Springer US, **2002,** 47-93.
[2] M Gray, Internet Growth Summary, *http://www.mit.edu/ people/mkgray/net/internet-growth-summary.html*, **1996**.
[3] RFC791, *Postal J Internet Protocol Darpa Internet Program Protocol Specification (IPv4)*, **1981** *https://tools.ietf.org/html/rfc*791.
[4] RFC2460, S Deering and R Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, **1998**, *https://tools.ietf.org/html/rfc*2460.
[5] S Hagen, *IPv6 Essentials*, O'Reilly Media, Inc., **2006**.
[6] P Gupta and N McKeown, Algorithms for Packet Classification, *IEEE Network Magazine*, **2001**, 15(2), 24-32.
[7] K Sklower, A Tree-Based Packet Routing Table for Berkeley UNIX, *In USENIX Winter*, **1991**, 93-99.
[8] T Kijkanjanarat and HJ Chao, Fast IP Lookups using a Two-Trie Data Structure, *IEEE Global Telecommunications Conference*, **1999**, 2, 1570-1575.
[9] MA Ruiz-Sanchez, EW Biersack and W Dabbous, Survey and Taxonomy of IP Address Lookup Algorithms, *IEEE Network Magazine*, **2001**, 15(2), 8-23.
[10] S Sahni and KS, Efficient Construction of Variable-Stride Multibit Tries for IP Lookup, *Proceedings of IEEE Symposium on Applications and the Internet*, **2002**, 220-227.
[11] F Baker and P Savola, *Ingress Filtering for Multihomed Networks*, RFC3704, **2004**.
[12] Fu J, O Hagsand and G Karlsson, Improving and Analyzing LC-Tries Performance for IP-Address Lookup, *Journal of Networks*, **2007,** 2(3), 18-27.
[13] R Rojas-Cessa, T Kijkanjanarat, W Wangchai, K Patil and N Thirapittayatakul, Helix: IP Lookup Scheme based on Helicoidal Properties of Binary Trees, *Computer Networks*, **2015**, 4, 78-89.