BEST LOOKUP ALGORITHM FOR 100+GBPS IPV6 PACKET FORWARDING

Suja G J[1], Sangeetha Jose[2]

[1]*M.Tech Scholar, Dept of Information Technology, Govt. Engineering College Painavu Idukki, Kerala, India*
[2]*Asst.Professor, Dept of Information Technology, Govt. Engineering College Painavu Idukki, Kerala, India*

Abstract

As internet traffic is growing day by day, it is a key requirement to increase more and more IP addresses. IPv4 cannot accommodate this need due to the exhaustion of its 2^{32} address. Hence in order to effectively satisfies this increase in demand for IP address new addressing scheme called IPv6 is developed. It has 2^{128} addresses so the exhaustion is not possible in near future. To forward IPv4 packet efficient mechanisms are there. But IPv6 is of 128bit length, so the present forwarding techniques is not possible to forward IPv6 packet effectively. In this paper we try to explore currently existing different lookup algorithms (Distributed Memory Organizations, Triec, Recursive Balanced Multi-way range trees and Range tree based IPv6 lookup) and analyze the best lookup algorithm for 100+ Gbps IPv6 packet forwarding by performing a comparative survey.

Keywords: *IPv4, IPv6, Lookup Algorithm.*

----------***----------

1. INTRODUCTION

We are living in an era in which, we always adopt best technology focus on how to achieve our goal as fast as possible. For that we always adopt best technology or development which saves our time. The emerging field like networking has a great influence on these developments. For the case of networking there are so many subsections like internet, mobile etc. Due to these developments the face of the world is also changed. In the case of internet the people always need to connect with their needs as much fast as possible. The packet or data from one place to another place must reach faster so that our need can be achieved smoothly. So our aim is to present best data forwarding or routing technique which will overcome the present disadvantage and improve the performance in data forwarding.

Modern societies rely heavily on information. If information was to stay in one place, then there would be no much use of it. This is what led to the vast development and employment of information networks, ranging from simple computer networks to advanced telecommunication networks. In any of these cases, information travels throughout the network and it is imperative to find its way from its source to its destination.

In communication network the information is divided into the capacity of the network as Maximum Transmission Unit and for the secure operation and for the transmission the data is fed in a **packet** with source and destination address and routed towards the destination through set of intermediate node. In each intermediate node it must be decided where to forward the packet based on a routing table and the destination. The destination address is looked up in the routing table to decide the action to be taken (see Fig-1). This process of searching in the routing table is called **address lookup.**

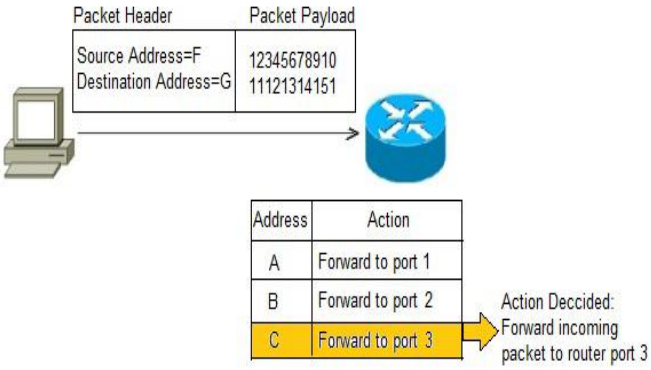

Fig-1: Router Lookup

The data or packet forwarding is the role of routers. So our work will focus on the router. So the router must forward packet to the destination address as much fast as it can possible by looking them to the destination address in the forwarding table of router it is called **packet forwarding**. There are so many techniques arises to achieve the role of mapping the destination address with the forwarding table of router this mapping process is called **lookup**[8].

Now our aim is that how we can achieve fast lookup. There are so many techniques available. We compare the lookup process and try to put forward a best technique. And we promise you that the new technique will give 100+ Gbps IPv6 Packet Forwarding on Multi-Core Platforms. For the explanation we need to familiar with some words now we move to the basics then can go to the solution.

1.1 Motivation

Internet traffic is growing day by day. So there is an increase in the need for more and more IP addresses. IPv4 cannot accommodate this need due to the exhaustion of its

2^{32} address. In order to effectively satisfies this increase in demand for IP address, new addressing scheme called IPv6 is developed. It has 2^{128} addresses so the exhaustion is not possible in near future. To forward IPv4 packet efficient mechanisms are there. But IPv6 is of 128bit length, so the present forwarding technique is not possible to forward IPv6 packet effectively. Hence, our motivation is to find a best lookup algorithm for 100+ Gbps IPv6 packet forwarding by comparing four present lookup algorithms: Distributed memory organization, TrieC, Recursive balanced multi-way range trees, Range tree based IPv6 lookup. This survey motivation is because of the increase in the IPv6 address usage. Now a days only 2% of the total IPV6 address is used but it gradually increases. Due to its increasing nature best lookup algorithm is an essential prerequisite to forward traffic as much speed as it can possible.

1.2 Organization of the Paper

We organized the paper as follows: In section 2, we glance into the preliminary concepts which are essential for the proper understanding of the related works. Session 3 deals with related work in this area. Session 4 elaborates existing various lookup algorithms and session 5 explains a detailed analysis with comparison. Session 6 concludes with an eminent open problem to be worked on.

2. DEEP LOOK ON THE PACKET FORWARDING

The goal of routers is to forward packet to their destination. When a router received a packet it is responsible to decide where to send the packet next. This decision can be either the packet destination or to the next router (**next-hop router**) which route it to the destination. If the decision is made then it also need to decide through which interface the packet is routed. For all these decision purpose the router will maintain a **forwarding table** or **routing table** which will store the information like the next hop address and the interface address. The router will gather this information with the help of **routing protocols** and the router will update its routing table periodically by these protocols. The router will consult the forwarding table each time when a packet is coming and use destination address as key to consult the forwarding table this process is called **address lookup**. On retrieving this information the router will forward packet from the incoming link to the appropriate outgoing link by the process called **forwarding** or **switching**.

2.1 IP Address

We always specified the word "address" throughout our discussion what really it is **IP Address** that means an identity to the devices on the internet. IP Address is the unique identity given to the device which when connected to an internet. The first address scheme come is the IPv4, which means it is a 32 bit IP address which is used to identify the devices. As the technology development increases the devices on the internet is increases. Therefore IPv4 address could not sufficient to address this issue. So a new address scheme is developed that is the **IPv6 address scheme**. It is a 128 bit IP address which will sufficient and we cannot go for a new scheme of IP address in near future. The difference between two address scheme is shown in the Table -1.

Table -1: Difference between IPv4 & IPv6 Address

	IPv4 Address	IPv6 Address
Deployed	1981	1999
Address Size	32 Bit	128 Bit
Number of Address	2^{32} (4,294,967,296) different address	2^{128} (340,282,366,920,938, 463,463,374,607,431, 768,211,456) different address
Address Format	Dotted Decimal Notation Eg: 192.168.10.1	Hexadecimal Notation Eg:1254:1532:26B1:CC14: 0123: 1111:2222:3333
Prefix Notation	192.168.0.0/24	1254:1532:26B1::/48
Address Scheme	Classful & Classless scheme	Classless scheme

Table -2: IPv4 Addresses

Class	Address Range
A	0.0.0.0 To 127.255.255.255
B	128.0.0.0 To 191.255.255.255
C	192.0.0.0 To 223.255.255.255
D	224.0.0.0 To 239.255.255.255 (Multicast)
E	240.0.0.0 To 255.255.255.255 (Future Research)

2.2 Classfull IP Address

From the Table -1 we find out that the class based IP address is IPv4. Here it divide the full ie. 2^{32} = (4, 294, 967, 296) IPv4 address space into five class of IP addresses which is listed in Table -2. But from the table we can understand that in class based schemes, there is IP address wastage happens. So the classless **subnetting** scheme is introduced. The subnetting divides the network into small network based on the requirements. In the case of subnetting packet is forwarded by **Classless Inter Domain Routing (CIDR).** The subnetting overcomes the IP address wastage. But due to the innovation of technologies the internet is growing to an unimaginable size so that the IPv4 couldn't sufficient so a new address scheme called IPv6 is introduced.

2.3 IPv6

As the requirements are growing faster, all IPs will be totally consumed soon. There will be no more available IPs. In order to solve this problem IPv6 was developed. The IPv6 scheme is represented in hexadecimal notation of 8 blocks separated by a colon (:). Every block contains 16 bits. This means that IPv6 scheme has 128 bit (ie. 8 * 16 = 128 bits). It has 2^{128} = 340, 282, 366, 920, 938, 463, 463, 374, 607,

431, 768, 211, 456 different address so we doesn't bother about the IPv6 address exhaustion in near future. Eg: 2001 : 0003 : 1b53 : 76ba : f678 : 8261 : 43bd : 3ab1 but storing this large value is memory wastage. We can also shorter this address by removing contiguous zero from every block (Eg. 2001 : 0000 : 0000 : 006b : 0000 : 0000 : 0874 : 4c32 as 2001 ::: 6b : 0 : 0 : 874 : 4c32).

IPv6 communication mode are :

- Unicast : One to one communication
- Multicast : One to many communication
- Anycast : One to the nearest communication. This means that we can give the same address to many devices, and the data that is sent from one sender will be delivered to only the nearest receiver found from those devices [2].

IPv6 special addresses are:

- '0:0:0:0:0:0:0:1' (Equals to '::1') Loopback address
- 'FF00::/8' Multicast address.
- '3FFF:FFFF::/32' and '2001:0DB8::/32' The range is reserved for examples and documentation.

2.4 Address Prefix

Due to the large value of IP address, to store it, we need a large memory space. Hence the best method is to represent a range of contiguous address by an address prefix. The prefix length is the decimal value specifying that the length of a prefix (i.e. the leftmost contiguous bits of prefix length are same). The prefix based address is written in the following format. The Fig -2 shows an example that we discussed.

"ipv6-address/prefix-length"

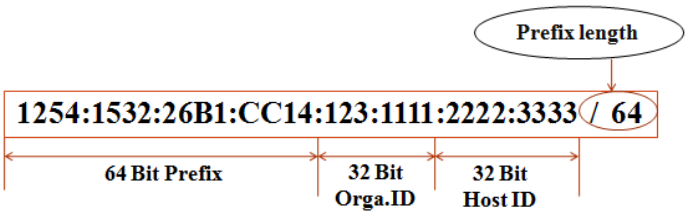

Fig -2: Prefix Notation

2.5 IPv6 Packet Structure

IPv6 datagram consist of header and payload data field and it is larger than IPv4 due to its increased in address as shown in Fig- 3. The header fields are listed in Table- 3.

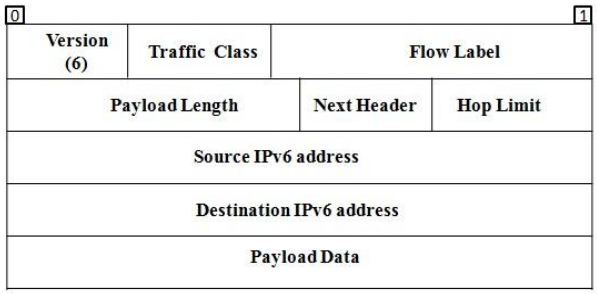

Fig -3: IPv6 Packet Structure

Table -3: Header fields of ipv6 packet structure

Field Name	No.of Bits	Indicate	Similar to IPv4 field
Version	4 bit	Version no.of IP (Consistency Check)	Version
Traffic class	8 bit		Type of service
Flow Label	20 bit	Datagram between source & Destination treated same	
Payload Length	16 bit	Length in bytes of remainder of datagram	
Next Header	8 bit	Protocol of the payload data	
Hop Limit	8 bit	Number of hops the datagram traverses	TTL
Source Address	128 bit	Initial sender of the packet	
Destination Address	128 bit	Address of intended recipient	

Table -4: Forwarding table

Destination Address Prefix	Next Hop	Outgoing Interface
1254:1532:26B1:::1324:1432:1342/64	2001:::6a:0:0:874:4c32/64	1
2154:1532:26C1:::1324:1432:1342/64	2001:::6b:0:0:874:4c32/64	2
1524:1532:26C1:::1324:1432:1342/64	2001:::6c:0:0:874:4c32/64	4

2.6 Migration from IPv4 to IPv6

Due to the development of IPv6 so many changes are there. However, there are many techniques that enable the network administrator to migrate from IPv4 to IPv6 without making any interruptions for the operation of the network.

2.7 Forwarding Table

The forwarding table is created in the router. The example for the forwarding table as shown in Table -4. By the lookup in the table we can forward packet to the destination. So that the table must be small as possible to decrease the searching time. The research shows that there are growth of routing table size due to the growing network.

2.8 IPv6 Packet Forwarding

There are different IPv6 packets forwarding techniques are there. Fig -4 is a representation of lookup processes. It select best lookup based on the following parameters:

- It should has low latency(Delay)
- High throughput
- Low memory requirements
- Should scale efficiently when the address width and/or the number of ranges increases (By scaling efficiently, we mean that an increase in address width and/or the number of ranges should affect).

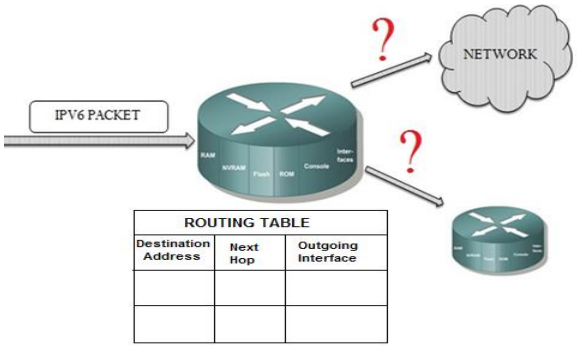

Fig -4: Router Decision Making

2.9 Requirements on Address Lookup Algorithms

It is important to review the characteristics of today's routing environment to derive adequate requirements and metrics for the address lookup algorithms. Using address prefixes is a simple method to represent groups of contiguous addresses [3]. Address prefixes allow aggregation of forwarding information and hence support the growth of the Internet. As shown above, the growth of a typical backbone router table. The characteristic of the routing environment includes:

- Search methods drastically reduce the search space at each step.
- Algorithms must be scalable with respect to the number of prefixes.
- Forwarding table needs to be updated dynamically to reflect route changes
- Backbone routers may receive bursts of route changes at rates exceeding several hundred prefix updates per second.
- The prefix length distribution in the forwarding tables can be used as a metric of the quality of the Internet hierarchy and address aggregation. Shorter prefixes represent a greater degree of aggregation. Thus, a decrease in average prefix length would indicate improved aggregation and hierarchy in the Internet.

3. RELATED WORK

The lookup algorithm is used to find the best outgoing links to the destination through which the packet is forwarded.

Distributed Memory Organization [7] is a novel algorithm and mutating binary search on hash tables organized by prefix lengths. This scheme scales very well as address and routing table sizes increase, requires a worst case time of $\log_2$(address bits) hash lookups and a Marker Storage algorithm for optimized storage.

TrieC[5] is a high-performance IPv6 forwarding algorithm. It is implemented efficiently in the Intel IXP2800 network processor (NPU). High performance can be achieved through close interaction between algorithm design and architectural mapping.

Recursive balanced multi-way range trees[6] it implemented a new category of prefixes covering prefixes and covered prefixes. Categorized the prefixes into hierarchy sets, prefixes overlapping would be no longer existent in a particular set. Naive binary search on prefix value would be applied on each set, search would adapt the PRM scheme and memory consuming is reduced because storage for precomputed information is free.

Range Tree-Based IPv6 Lookup[1] Tree-based solutions, are elegant in terms of the number of memory accesses required to perform a lookup, which is O(log N), where N is the number of keys/ranges. Here a tree of range based lookup algorithm is presented.

4. DETAILED LOOK ON LOOKUP ALGORITHMS

The migration from IPv4 to IPv6 addressing is gradually taking place with the exhaustion of IPv4 address space. This requires the network infrastructure to have the capability to process and route IPv6 packets. However, with the increased complexity of the lookup operation and storage requirements, performing IPv6 lookup at wire-speed is challenging.

The different lookup algorithms taken for comparison are:

1) Distributed memory organization[7]
2) TrieC[5]
3) Recursive balanced multi-way range trees[6]
4) Range tree base IPv6 lookup[1]

4.1 Distributed Memory Organization [7]

Distributed Memory Organization is an IPv4 & IPv6 novel lookup algorithm. Based on prefix length a hash table is used and uses a mutating binary search on this hash table it requires $\log_2$ (address bits) of worst case hash lookup time. It has 5 hash lookup for IPv4 and 7 hash lookup for IPv6. It scales very well when address and routing table size increases. Optimized storage is achieved by Marker Storage algorithm. This is achieved by classifying the addresses stored in the routing table by analyzing the data of prefixes.

Table -5: Memory module allocation

Lookup unit No.	Bits 1,2,3,4
1	0001
2	0010
3	0011
……	……
16	1111

In Distributed Memory Organization prefixes are stored in a routing table. Depending on certain bits the routing table is classified into several flows. If 4 bits are used as ID bits and based on this ID bits routing table is classified into 16 categories as shown in Table -5. If the prefixes are less than 4 then it is expanded to 4 bits for example 110* is expanded.

The destination IP address is classified into 16 categories based on four ID bits. Then longest matching prefix is find out by using the parallel lookup algorithm 1. The complete parallel lookup mechanism and the distributed memory organization is shown in Fig -5.

4.2 TrieC[5]

Trie is also an IPv6 lookup algorithm. It mimics the features of the multicore and multithreaded system. As from most analysis it shows that tree (trie) based techniques is used to find a longest prefix match from root node to a matching leaf node. It ignore 1st 3 and lowest 64 bits, build a tree which cover prefix with length between 3 and 49 and all others are find out by hashing.

Algorithm 1: Parallel lookup Algorithm

1. **Function** Lookup (Destination Address).
2. Use the ID bits of Destination Addresses to classify.
3. Push the IP Addresses into the FIFO of the corresponding Lookup unit.
4. **For** each Lookup unit simultaneously do
5. **While** (FIFO not empty) do
6. Pop an address from the local FIFO.
7. Use binary lookup schemes to find BMP.
8. Push the Next-Hop-Address into Output cache.
9. End **While**
10. End **Loop**
11. End **Function**

It uses a modified compact prefix expansion technique in which routing table consist of duplicate next hop so it must be compress the bit and store it only once. Next-hop indices A appear 48 times and B appear 16 times in a block therefore these bits are made only in next hop index array with ABA made once. The lowest 18 bits in a 64bit address prefix are called Tindex and lowest 6 bits are used as another index to search a bitAtlas which locate correct next hop index NHIA. The IPv6 TrieC lookup algorithm is given in 2.

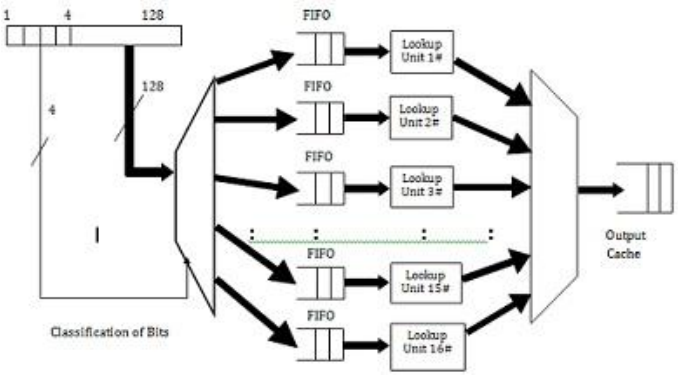

Fig -5: Distributed memory organization and parallel lookup

4.3 Recursive Balanced Multiway Range trees [6]

Recursive Balanced Multiway Range trees **(**RBMRT) is a tree based IPv6 lookup algorithm. Here there are two classes of algorithms covering and covered prefixes. Covering prefix again subdivided into level-1 and level-2+ covered prefixes. RBMRT mechanism consist first a data structure design and then the lookup procedure. It is also a tree based structure in which it is divided into different hierarchy. Each hierarchy holds the pointer to the next level hierarchy ie, first level remember the 2nd level hierarchy. Routing table is shown in Fig -6. Data structure is shown in Table -6.

Algorithm 2: TrieC Lookup Algorithm

```
IPv6_Lookup_TrieC(IN DstIP,OUT Next-HopID)
1.      Current_Block=TrieC15_6;
2.      Tindex=DstIP[124:110];
3.      Bit_Vec=GetBitVec(Current_Block,Tindex);
4.      BAindex=DstIP[109:104];
5.      NHI=GetNHI(Bit_vec,BAindex);
6.      if(NHI.flag=0) return NHI.Next-HopID;
7.      else{ // search TrieC4/4 tables, base[i] is base of
8.      (i + 1)th-level TrieC tree
9.      Current_Block=TrieC4/4 at Base[0]+NHI[14:0];
10.     for( i=1;i¡=3;i++ ) {
11.     Tindex=DstIP[103-8*(i-1):100-8*(i-1)];
12.     Bit_Vec=GetBitVec(Current_Block,Tindex);
13.     BAindex=DstIP[99-8*(i-1):96-8*(i-1)];
14.     NHI=GetNHI(Bit_Vec,BAindex);
15.     If( NHI.flag=0) return NHI.Next HopID;
16.     Else {
17.     if(i!=3) Current_Block=TrieC4/4
18.     at Base[i]+NHI[14:0] << 4;
19.     else break; //search longer prefix in Hash16
20.     }
21.     }
22.     if(Hash(DstIP[79:64])) return Next-HopID;
23.     else return Default-Next-HopID;
24.     }
25.     } // IPv6 Lookup TrieC
```

If the destination address address for the lookup is fall in the range P6 then the lookup operation is carried in the order first it look in the Btree which is the root and which contain all the 0th hierarchy prefix so get the first prefix P1 and it then point to the prefix set P4 and it again reach set p and there is no more remember in p6 so the longest prefix is p6 and it is returned.

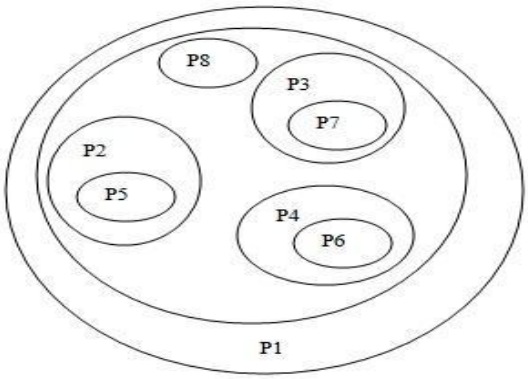

Fig -6: Example of prefix set

Table -6: Data structure of prefix

P1	1*
P2	100*
P3	101*
P4	110*
P5	10000*
P6	10001*
P7	10101*
P8	11001*

4.4 Range Tree-Based IPv6 Lookup [1]

Tree-based solutions are elegant in terms of the number of memory accesses required to perform a lookup, which is O(log N), where N is the number of keys/ranges[1].

Enabling Parallelism for IP lookup: Parallelism means perform computation in parallel. It is essential for high performace and it may exploit the multiple processing capability of GPP's.

The most basic method of parallelization can be thought of as **search tree duplication**, in which case, all partitions possess the entire routing table information and lookup can be carried on independent of other packet lookups. However, this causes the memory required to store the search tree to increase proportional to the number of partitions. While this is not a desirable solution, in most cases on software platforms, this translates to increased lookup time since the cache memory will not be sufficient to store the duplicated search trees, especially for large routing tables.

A more attractive method to perform partitioning is to place the prefixes in a set of bins in such a way that the prefixes in one bin are disjoint from those in other bins. In the context of our problem, IPv6 lookup, the state of being disjoint can be described as being able to search in only one bin and finding the corresponding routing information, without consulting the other bins. Since packets correspond to different prefixes in a given trace, when more disjoint partitions are present, the more likely for them to be processed independently and in a parallel fashion. This provides opportunities for parallelism. However, it is imperative that the formed partitions are of similar sizes. Otherwise, it gives rise to various other issues such as fairness (some packets experiencing longer latency than others) and uneven memory distribution.

Disjoint Partitioning: Disjoint partitioning can be achieved in numerous ways. Using

- The leftmost bits of the IP addresses
- A selected set of bits from the IP addresses can be chosen as well.

The initial bits based partitioning divides the address space into multiple disjoint sections and considering subsets of prefixes belonging to each section as a smaller routing table. If p bits are used, there can be as many as $O(2^p)$ partitions, depending on the prefix distribution of a given routing table. Partitions formed using this method can be of disparate sizes. Also, the prefix with length less than p needs to be expanded under this partitioning scheme.

Disjoint partitioning has two main benefits:

1. Identification of the corresponding partition for an incoming packet is simply a table lookup which can be completed in O(1) time and
2. Even though the initial partitioning may not be balanced (i.e. near-uniform prefix distribution across partitions), it is relatively easy to achieve balanced partitioning by aggregating initial partitions.

Algorithm and Partitioning: After partitioning using the p leftmost bits is done, process them further to achieve balanced partitioning. The rationale behind this further processing is that the initial partitions may contain different number of prefixes which causes the packet latency for different partitions to change significantly across partitions. Hence, perform an aggregation operation in which subsets of initial partitions are combined to form a single aggregated partition. This step may not ensure perfectly balanced partitioning depending on the distribution of the prefixes. The aggregation algorithm can be described in Algorithm 3.

For a routing table with N prefixes, time complexity for initial partitioning is O(N). To form the aggregated partitions, the time complexity is $O(n_i)$ where n_i is the number of initial partitions formed, hence the time taken to form the partitions is fairly small. The number of bits used for partitioning is denoted by p, and n_i and n_a stand for initial and aggregated number of partitions, respectively. varied p and set maxsize to the largest initial partition size of each scenario. One can use the tuning parameters p and maxsize to adjust the number of aggregated partitions (n_a) created.

Lookup Algorithm: Now we discuss about how to map the partitions formed by the aggregation algorithm onto multicore platforms and how the packet forwarding is performed. The initial portion of the lookup is the partition identification.

Algorithm 3: Aggregation Algorithm

1. Define the maximum size for the aggregated partitions, maxsize
2. If all partitions are marked used go to step 8, if not, go to step 3
3. Select the largest initial partition that is not marked used
4. If the selected partition size is equal to or greater than maxsize, mark the selected partition as used and go to step 2
5. If all partitions are marked used go to step 8, if not go to step 6
6. Starting from the smallest partition that has not been marked used, combine the prefixes from the smaller partition into the larger partition

7. If the aggregated partition size is equal to or greater than maxsize go to step 2, otherwise go to step 5
8. Algorithm complete

This can be simply realized using an initial lookup table. For example, If p bits are taken then make lookup table of size 2^p is created and sub pointer information of each partition is stored in it. O(1) is the search complexity due to the aggregation step.

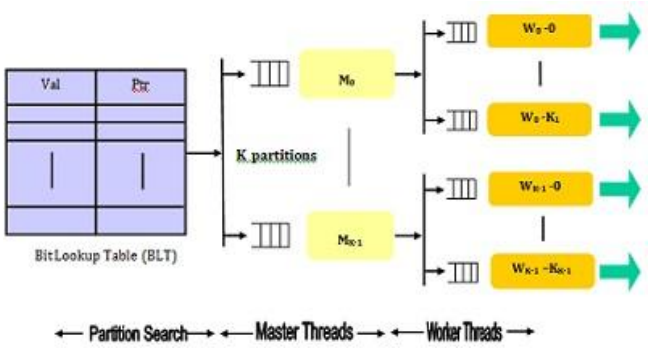

Fig -7: Hierarchical multi-threaded architecture of the proposed IPv6 lookup engine [1]

We can adopt Master-worker architecture. Here when the partition search is completed the packet is forwarded to the corresponding Master M_i and which create the worker thread W_{i-j}, where j is worker thread number. It is flexible architecture because the worker thread creation can be controlled by the user. The basic architecture is shown in Fiig -7.

On multi-core platforms, it is desirable to have more partitions (i.e. more master threads) which provide more opportunity for parallelism. Further, when the number of partitions is higher, the number of prefixes (hence the height of the range tree) per partition decreases. This effectively reduces packet lookup latency. Also, the explicit range tree approach is more suited for the software engine since the search can be terminated when the corresponding node is found. This comes at the cost of higher memory consumption. However, on GPPs, this is a minor concern.

5. ANALYSIS OF LOOKUP ALGORITHMS

From the survey, we can summarize about different lookup algorithms as as shown in Table -7. It shows that the Distributed Memory Organization require time complexity of O(log (log N)) and in TrieC it is $\log_2$ (address bits) for RBMRT it is $O(\log_k N)$ and for the range tree it is O(N). The complexity parameter comparison shows that the search and time based complexity is best for the range tree based solution.

Comparison also shows that other method can scale very less but the range tree based solution can perform high lookup of 2 million packets at a time. This will overcome the disadvantages of all existing lookup algorithm especially for the IPv6. The proposed solution will exist for the development and to overcome the challenge of the growing of the IPv6 and the scheme provide 100+ Gbps IPv6 packet forwarding.

Table -7: Complexity based analysis

Lookup Method	Complexity	Concept
Distributed Memory Organization	Time=O(log(Log N))	Table Partitioning
Triec	Time=$\log_2$ (address bits)	Based on the Longest prefix match
Recursive Balanced Multi-way Range Trees (RBMRT)	Memory=2 N Search=O(log N) Time=O(log N)	Based on a novel data structure
Range tree Based IPv6 Lookup	Memory=O(log N) Search=O(1) Time=O(N)	Routing Table Partitioning based on range

6 CONCLUSION AND FUTURE DEVELOPMENTS

We have conducted a literature survey on the existing lookup algorithms and it is summarized as shown in Table -7 and Table -8. It shows that the range tree it is best in time, memory and lookup performance complexity. Feature based comparison also shows that other method can scale very less but the range tree based solution can perform high lookup of 2 million packets at a time. This will overcome the disadvantages of all existing lookup algorithm especially for the IPv6. Range tree based IPv6 lookup is more suitable especially for multi-core platform. Updation is also limited to only one partition and hence updation performance is also very high. Experiment results shows that range tree based lookup has 3 times lower the memory consumption as that of Triec and has 5 times more lookup rate than the corresponding RBMRT.

Table 8: Feature based analysis

Lookup Method	Advantages	Disadvantages
Distributed Memory Organization	Design simplicity Scales very well when RT sizes increase	Can perform only 16 lookup at a time
TrieC	Thread synchronization Latency hiding	Memory bottleneck occur at some Time
Recursive Balanced Multi-way Rang Trees(RBMRT)	achieve the optimal lookup time or binary search	Search Time Increases

Range tree Based IPv6 Lookup	2 Million packets processed at a time memory consumption is 3 times lower than that of triec 5 times higher LOOKUP rate than that of RBMRT	

We can also suggest that performance of IPv6 packet forwarding can be improved drastically by combining the concepts of range tree based IPv6 lookup with that of the recursive balanced multi-way range tree and this is an eminent open problem to be worked on.

REFERENCES

[1]. Thilan Ganegedara and Viktor Prasanna Ming Hsieh, "100+ Gbps IPv6 Packet Forwarding on Multi-Core Platforms", Dept. of Electrical Engineering University of Southern California Los Angeles.
[2]. Mohamed Salem Salem Ali Easa, "CCNA in 21 Hours", bookboon.com
[3]. Miguel A. Ruiz-Sanchez, Ernst W. Biersack and Walid Dabbous, "Survey and Taxonomy", January 15, 2001
[4]. Hoang Le and Hoang Le, "Scalable Tree-based Architectures for IPv4/v6 Lookup Using Prefix Partitioning".
[5]. Xianghui Hu, Hefei and Xinan Tang, "High-performance IPv6 Forwarding Algorithm for Multi-core and Multithreaded Network Processor".
[6]. Pingfeng zhong, "An IPv6 Address Lookup Algorithm Based on Recursive Balanced Multi-Way Range Trees with Efficient Search and Update".
[7]. Pankaj Gupta, Uma Nagaraj, Nikhil Anthony, Deepak Jain, Pranav Gupta, and Harsh Bhojwani, "Advanced Routing Algorithm for IP Lookup (IPv6)",
[8]. Georgios Stefanakis, "Design and implementation of range trie for address lookup".

BIOGRAPHIES

Suja G. J currently doing post graduation in network engineering in Department of Information Technology at Government Engineering College Idukki. Her area of research includes networking software engineering, cloud computing & cryptography. Suja G J can contacted at **gjsujavipin@gmail.com**

Sangeetha Jose is working as faculty in the Department of Information Technology at Government Engineering College Idukki. Her areas of research interests mainly focus on cryptography, network security, security issues in cloud computing, computer networks, software engineering and algorithms. Sangeetha Jose can be contacted at **sangeethajosem@gmail.com.**