Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2016, 3(10): 32-35

Research Article ISSN: 2394 - 658X

Hash Data Structure for IPv6 Filters

Rohit G Bal

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

ABSTRACT

IP filtering is a technique used to control IP packets flow in and out of a network where Filter engine inspects at source and destination IP of incoming and outgoing packets. Here Filter engine is designed to improve the performance of the filter, i.e. to reduce the processing time of the filtering mechanism. The data structure used in the IP filter is hashing, for larger number of hosts and various ranges IP network of hosts hashing provides much better performance than link list. Here hash function for the hash table is valid IP ranges. In hash table technique the comparison can be done with minimum number of comparisons.

Key words: IP Filter, Networks, Firewall, Hashing, Hash Table

---

INTRODUCTION

In these days Wide Area Network (WAN) becomes backbone of the communication for various fields like business, government agencies, corporation, education, banking, etc. The internet is growing continuously and the availability of sophisticated attacking tools makes internet a very dangerous place. 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) consists of numerous remote host access system with help of network. With this remote accessing often there is unauthorized access to network resources [3]. These unauthorized accessing sometimes which would cost us tremendously. To avoid these, we have to protect several services from several host in WAN.

IP filter is mechanism that keep the unwanted/unauthorized remote accessing at bay with help of set of rules implied by the user [3]. Rule can be set for either allow or deny or both. Rule also can be set for ingress or egress. For storing the rules in the system there will be a table in the memory. IP Filter will check all IP packets going through it i.e. both incoming and outgoing packets and compare the source and destination IP.

Allow - Allow the packet to pass in/out of the network
Deny - Deny the packet to pass in/out of the network
Ingress – Packet coming inbound from outside network to protected network (inbound traffic)
Egress - Packet going outbound from inside network to unprotected network (outbound traffic)
* means all IP

In IP Filter Engine there is a table which stores the set of rules decided by the user. According to the rule the packet is analysed and sent to/from the network by the IP filter. The filter analyses each packet going through it. IP filter will compare the packet and the set of rules stored in the memory. After comparison filter will decide to allow or drop the packet. The rules of the filter are stored in data structure in memory as shown in Table -1.

The paper is organized as follows: Section 2 describes about IP. Sections 3 Hashing Tables & hashing functions, IP Hashing Filter Architecture. Section 4 presents related results. Section 5, we draw the overall conclusions.

Table -1 Table that Stores the Rules of IP Filter in Memory

IP	In/Out	Allow/Deny
*	Egress	Deny
*	Egress	Allow

INTERNET PROTOCOL VERSION 6 (IPv6)

IP version 6 (IPv6) [4] is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4) [5]IPv6 solves the address depletion problem of IPv4. In IPv4 maximum number of IP that can be used is up to 2^{32} while in IPv6 it is up to 2^{128}. 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 1023 addresses are available for every square meter of the Earth's surface [5].

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 [6].

- 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.

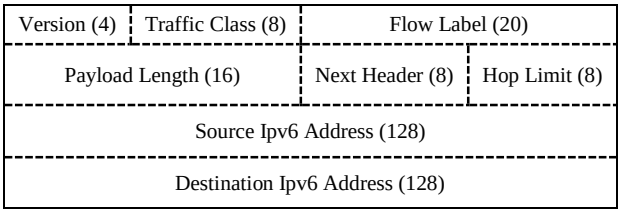

Fig. 1 IPv6 Packets Header Format

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 [6].

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 nodes address those 2 nodes are in same network which needs no routing. And if M+N is different routing is necessary. These M+N bits is like Network bits in IPv4 address [6].

HASH TABLE DATA STRUCTURE

Data structure is an organized way of storing data to use the data more efficient ways in memory for operations. Rules of filter engine is stored in form of data structure in the memory. These rules are used to compare the source /Destination Address stored in header of IP packets entering or exiting the interface. Hash table is an associative data structure [7].

Ordinary array uses technique of direct mapping; direct addressing is applicable when we can afford to allocate an array with one position for every possible key. Hash table is generalization of array, here elements are stored in size

proportional keys instead of independent key for each position. Consider N different elements for searching for array and linked list there is N no of possible keys i.e. each location may be key. But in Hash table the N keys are generalized called buckets according to hash function defined. Elements are grouped under a function in hash table called hash function. In real time, compared to array and linked list for searching hash table gives much greater performance in terms of time complexity, because of for finding the key program doesn't have to traverse entire set of elements. Hash function will direct the searching algorithm to the generalized group of elements. Given below figure 4 compare possible number of keys of data structure [7-8].

Table -2 Comparison of Data Structure

Data Structure	No of Elements	Possible Number of Keys
Linked List	N	=N
Hash Chaining[#]	N	<=N

- Possible number of keys depend on the hash function used, it determines the size of bucket.

Hash Function

Hash function maps the keys to the bucket where the key belongs. Hashing function determine the number of buckets created for inputs. Ideal hashing functions should be designed to have similar bucket size for all bucket. There are many hash function available in the according to the input, number of input it can be chosen. Here the hash function used is IP address according to the range of IP address which will allow to create 4 buckets of IP in the hash function [7] [9].

The source and destination IP ingress and egress packets are compared with the values in the hash table. Based on the range of IP address hash function will choose the bucket for comparison of the packets IP address and the IP address stored in the bucket. According to the rule defined in the filter the packet is forwarded or dropped. The real time IP provide a faster way of packet processing in the IP filter which will avoid the latency of lookups in the linear database.

Requirements for Algorithms

The following are the observations made as requirements for the selection of hash function

- Storage Efficiency for the IPv6 address
- Should able to store variable length prefixes
- Dynamic insertion and deletion for dynamic routing table updates

RESULTS AND COMPARISON

For finding the results and comparison we are considering the following data structures [8], [10-12] **linked list**, **hash chaining**. There are 3 main factors that has to be considered before choosing data structures. They are space, time and simplicity

a) Space

The amount of memory required to store a value should be minimized. This is especially true if many small nodes are to be allocated.

b) Time

The algorithm should be efficient. This is especially true if a large dataset is expected. There are 3 cases to be considered, best case and worst case.

c) Simplicity

If the algorithm is short and easy to understand, fewer mistakes may be made. This not only makes your life easy, but the maintenance programmer entrusted with the task of making repairs will appreciate any efforts you make in this area. Here we are calculating the easiness by calculating Lines of Code (LOC).

Comparison Based on Theoretical View

For comparison of the memory we are considering the extra amount of memory required for the pointers for large number of input. For hash tables, only one forward pointer per node is required.

In addition, the hash table itself must be allocated. Considering the size of number of IPs stored we can avoid the size of hash tables. For linked lists, only one forward pointer per node is required. The comparison between Linked List and Hash Table is shown below [7], [10-12].

Because Real time IP filter focus more on the searching time of IP inside the data structures the Hash table gives better performance than the Linked list. Even though searching algorithm is considerably complex to implement than linked list. Because of the performance given for searching is consider important, hash table will be good choice.

Table -3 Comparison of Pointer Memory Required

Data Structure	Elements	Pointers
Linked List	N	N
Hash Chaining	N	N

Table -4 Comparison of Time Complexity

Data Structure	Average Case	Worst Case
Linked List	Θ (n)	Θ (n)
Hash Chaining	Θ(1)	Θ (n)

CONCLUSION

The above suggested data structure provides much better performance than the linked list. In linked list searching take time because for searching the element in last place algorithm has to go through all the other elements before reaching solution. In case of the hash table will divide the data structure into buckets of smaller linear data structures and only have to search lesser number of elements. In future more effort is needed to select the hash function or other data structures like tries [13] can be used for faster packet classification in IP filter.

Acknowledgements
I would like to thank the anonymous reviewers for their helpful suggestions on the organization of the paper.

REFERENCES

[1] KG Coffman and AM Odlyzko, *Internet Growth: Is there a Moore's Law for Data Traffic?* Handbook of Massive Data Sets, New York: Kluwer, **2002**.
[2] M Gray, Internet Growth Summary, MIT, 31 January 1996. [Online]. Available: *http://www.mit.edu/people/mkgray/net/ internet-growth-summary.html*., **2016**.
[3] B Corbridge, R Henig and C Slater, *Packet Filtering in an IP Router*, San Diego, CA, LISA, **1991**.
[4] S Deering and R Hinden, *Internet Protocol Version 6 (IPv6) Specification*, RFC, **1998**.
[5] JB Postel, *Internet Network Protocol Specification*, Version 4, RFC, **1978**.
[6] S Hagen, *IPv6 Essentials*, CA: O'Reilly Media, Inc, **2006**.
[7] R Stephens, *Ready-to-Run Visual Basic Algorithm*, New York: John Wiley & Sons, **1998**.
[8] TH Cormen, CE Leiserson and RL Rivest, *Introduction to Algorithms (2nd Ed.), Stein*, Clifford: MIT Press and McGraw-Hill, **2001**.
[9] RG Bal, IP Packet Filtering using Hash Table for Dedicated Real Time IP Filter, *International Journal of Wireless and Microwave Technologies*, vol. (In Press), **2016**.
[10] TH Cormen and E Charles, *Introduction to Algorithms*, 2nd Edition., New York: McGraw-Hill, **2009**.
[11] DE Knuth, *The Art of Computer Programming*, Volume 3, Sorting and Searching, Reading, Massachusetts: Addison-Wesley, **1988**.
[12] PK Pearson, Fast Hashing of Variable-Length Text Strings, *Communications of the ACM*, **1990**, 33 (6), 677-680.
[13] RG Bal, Review on Tries for IPv6 Lookups, *European Journal of Advances in Engineering and Technology*, **2016**, 3 (7), 28-33.