International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT228623

# Splay Tree and Skip List: Effectiveness and Analysis in Internet Packet Categorization

**Prof. Sumit S Shevtekar, Harish Sumant, Abhijit Suryawanshi**
Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India

**Article Info**

**Publication Issue :**
Volume 8, Issue 6
November-December-2022

**Page Number :** 285-294

**Article History**
Accepted: 12 Nov 2022
Published: 28 Nov 2022

**ABSTRACT**

Internet packet traffic has expanded rapidly as a result of the rising number of world wide web users and the amount of information sent by software applications which has highlighted the necessity of speeding up the processing required in network systems. Packet categorization is among the strategies utilized across network architecture. Skip List and Splay Tree, two of the most important data structures used in decision trees, will be investigated for their effectiveness in packet categorization in this study. The period of packet identification, the amount of memory accesses, and memory consumption per event make up our performance criterion. These criteria are used to judge whether data structure, between Skip List and Play Tree, is the preferable one to use for accurate packet classification.

**Keywords :** Skip list, Splay tree, Decision Tree, Packet classification, Data Structures

## I. INTRODUCTION

Internet packet traffic has substantially increased as a result of the rise in users and the volume of information that programs are exchanging. For this reason, a crucial proce- dure called packet classifications is applied to speed up the processing necessary in network systems such as routers.The broadest packet-switching network is the Internet. In this network, data is sent from the source to the destination in the packet form.The various streams of packets in network systems are termed to as ”network packets” by classifica- tion.Number of network methods incorporate packet routing, packet classification, and packet guiding policies.using these key principles, packet flow processing has become possible at very high velocities, and the same rules may well be applied to every local traffic policies.Applications functioning over such a network that involve packet classification contain security operations, traffic management and quality of service (QOS), and policy-based routing are the three distinct areas.Numerous research have benchmarked various packet detection methods analytically or operationally.

### A. Desorption

Two processing phases are used by a decomposition-based method. In the first stage, each filter is searched separately, and in the second step, In the second stage, the intersection is used to combine the results of the all the searches on the various fields. As a

result, the method has a lot of parallelism potential. Memory use is high due to the big amount of the data.

B. Extensive search

The primary drawback of these algorithms is the linear relationship between time complexity and the quantity of filters. In this kind of method, each item in a list is examined to see if it matches the search query argument.

C. Decision tree

Based on the binary patterns in the prefix fields of the filters, these techniques store the set of filters in search trees. In order to create a decision tree based on many fields, a tree is constructed in which the leaves have a defined filter or a collection of filters that intersect the traversed prefix from the root to the leaves. These algorithms examine the binary contents of the relevant fields on the search tree to find the most suitable filter matching the input package

The current approaches are not balanced in terms of time and memory usage. Binary trees, on the other hand, function well when the elements are introduced unintentionally but degrade when the procedures are carried out sequentially. Different data formats are used by tree algorithms when searching. The splay tree and the skip list are two of the most significant data structures commonly used in decision trees.

The history of access to a splay tree's elements determines how well it performs. The effectiveness of a skip list, on the other hand, is based on an independent randomization of the height of links that point to particular parts. Therefore, the operation of splay trees and skip lists is examined using probabilistic approaches. For a probabilistic examination of the complexity of these algorithms, we point the reader to the references.

Using these two different data structures, we will assess and contrast the performance of packet-classifying tree algorithms in this work. We'll apply the temporal complexity and memory complexity standards for this. Memory complexity depends on how much memory is required by the method's data structure.

Time complexity depends on how many memory locations are used by the algorithm to categorise each packet.

The paper is divided into the following sections for struc- ture. Firstly, we review the history of packet classification tree algorithms and the associated earlier efforts in order to assess the performance of these algorithms.Fundamental design and implementation of tree algorithms based on skip lists and splay trees is discussed in the third section. A fourth section explains the tools for creating filters and packets and compares the performance assessments in two ways after explaining the evaluation criteria. A conclusion is presented in the concluding section, along with suggestions for future research.

D. Tuple spaces

Partitioning of filters is done by the count of bits supplied in the search query's prefixes, splitting the search space into distinct sub searching areas.Matching and verification of the incoming packets is done at the time of classification upon the prefix fields of interest using basic and tree-based search algorithms.The packet which is matched with the tuple is successful ,the tuples that are equivalent to sets of the tuple with regard to their matching with other fields of the packet those filters are evaluated. The memory complexity is more efficient then decomposition based algorithm.It is determined whether these filters are consistent with some other elements of the packet by comparing them with sets of tuples.

E. Background

Examining the performance of the skip list and splay tree data structures when used in multidimensional search- ing on a packet classifier's rule set is the major goal of this work.Because of the manner such data structures are examined, inserted, and updated, tree-based packet analyzers can minimise the amount of memory required throughout the search and, as a consequence, the complexity of categorization. According to a review of prior research, no study has been conducted to properly evaluate the effectiveness of packet- classifying tree algorithms based on skip lists and splay trees. Previous attempts did not compare the effectiveness of various algorithms; instead, they concentrated on optimizing them.

The skip list was used to increase the speed of data gathering techniques in local lists. The skip list is designed to start the search with the nearest node previously retrieved from this prefix. As a result, tremendous time is saved. Comprehensive tests show that their approach can outperform the original design on a 4 byte device.

To improve the performance of the firewall, Trabelsi et al. (2015) introduced a multi-stage and dynamic packet filtering system in 2015. Splay tree filters are used to implement their suggested mechanism, and it makes advantage of traffic features to cut down on packet filtering time. The most favorable customized pattern for the tree and evaluate if dynamic updates of the Splay Tree can be predicted by it. Network window traffic is filtered by the filters. Initial acceptance of the packet is performed using the splay tree

data structure which is dynamically updated in line with the network traffic streams. As a result, frequent packets require less memory access, which cuts down on the time required to filter all of the packets.

Zhong, Geng, and Zhao (2013) concentrated on a straightforward yet significant type of remote authentication challenge. Membership requests are validated by this form which has a collection of dynamic n data components which are stored in folders with no name. They examined certain membership request confirmation approaches which were already existing. Few examples are Merkle hash tree, skip list and RSA tree. The data structures that the algorithms in each of these methods utilise to update the data are either too slow or have a high level of complexity. Various data structures may also be able to be rebuilt throughout the updating process. B+ trees having RSA accumulators are chosen for the authentication mechanism. Low processing costs for membership queries are properties of B+ trees hence they are used.

An incoming packet can be quickly rejected by maximising the comparison order of the matched security-rule fields. Trabelsi Zeidan (2012) established a mechanism in 2012 to enhance the filtering time of firewall packets. Their suggested solution relied on rearranging the filtering fields in accordance with traffic data. It also permitted the use of classifiers with many levels. Consequently, their suggested approach might be seen as a defence against denial-of-service attacks for the device (DoS). Splay trees are used to achieve early packet acceptance, and they alter dynamically in relation to traffic streams. An improvement to the prior method known as Self-adjusting Binary Search was sought. The prior method had several drawbacks that were addressed by the proposed method. Frequent packets require less memory access. Their approach therefore according to the simulation results, their proposed mechanism can improve the firewall's performance in terms of total packet processing time, in contrast to the SA-BSPL approach.

Zeidan Trabelsi (2011) presented a method in 2011 to improve firewall performance by thwarting denial-of-service assaults. They accomplished this using a statistical traffic plan that was implemented as a multi-level filter, splay tree, and hash tables, along with a security policy of filtering. The suggested approach uses less RAM since it quickly discards undesirable traffic and repeating packets. In general, packet matching time is decreased as a result. The examination of this method's results shows that the proposed mechanism greatly shortens the time required to process DoS traffic.

F. Algorithms and tools

The algorithm is described in this section. Consider the illustration rules in Table 1. If the source addresses are equal, the sorting operations are carried out in accordance with the destination addresses. A list of criteria is provided here that can be used to assign priorities to source addresses based on the length of the addresses. The address at the top of the list is given priority over all others because the criteria are ordered from highest to lowest based on address length (Trabelsi Zeidan, 2012). Source and destination prefix addresses must be converted into a list of numbers.The higher and lower limits for each prefix are listed in Table 2 for this purpose. The prefix addresses are displayed in a convenient six-bit format.

1) Splay Tree: Every splay tree should be created for a field, which includes the protocol type, source address, destination address, source port number, and destination port number. The tree nodes contain values, counters to count packet matches, lists of rules, and pointers to child nodes and parents. All nodes' counters are initially set to zero.Each node in the protocol tree contains a list of rules with protocol fields that are equal to the node's value, but nodes in other trees contain lists of rules with lower boundaries that are less than or equal to the node's value and upper boundaries that are greater than or equal to the node's value.The source address, destination address, source port number, and destination port number values should be added to the respective trees in two phases since they have higher and lower bounds.

The lower boundary is set in the first phase. A node with a lower boundary value than the root value is placed on the left tree, and a node with a higher boundary value than the root value is placed on the right tree. Each rule is then evaluated by comparing its upper and lower boundary values to the value of the lower boundary node. The rule's ID is added to the list of lower boundary rules whenever the value of the lower boundary falls inside the range of that rule. The rotation operation will be used to relocate the lower boundary node to the tree's root after it has been added to the tree.Inserting the higher border into the tree is the second step. This action is comparable to inserting the lower boundry.

Any list or tree's search results come with a list of matching rules. An intersection operation is carried out between the five lists acquired from the splay trees in order to identify a common rule between them.

Figure 1 illustrates the procedure for making a splay tree for the source address fields of the rules in Table 2. The R1 rule has been added to the tree in Fig. 1A. In order to do this, the lower boundary value is first entered. The rule with ID 39 is placed at the base of the tree after R1 and R5 have their IDs added to the rules list, since 32 falls between them. The R2 rule has been incorporated into the tree in Fig. 1B. In this instance, the value 16 is added.The IDs of R2, R3, and R5 are added to the node's rules list after the node 16 has first been looked for in the tree. The node 16 will be correctly positioned if the data is missing. The correct rotations between numbers 23 to 39 and 16 to 32 are used to finally move node 16 to the root.The number 23, which represents the bottom border of R2, is added to Fig. 1C. The R2, R3, and R5 rules are

added to its set of rules in the next phase. After that, the node 23 is moved to the root by rotating it left between the nodes 23 and 16 and right between the nodes 23 and 32.The R3 rule is subsequently added to the tree. No modification to the tree takes place because its values have already been inserted.

**Table 1** **An example of a set of rules.** Each row of the table is a rule of a rule set. Each rule is represented as constraints on source IP, destination IP, source port number, destination port nummber and protocol.

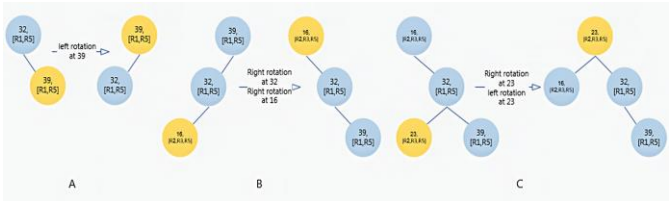
| Rule | Source IP address | Destination IP address | Source port number | Destination port number | Protocol |
|---|---|---|---|---|---|
| R1 | 100* | 011* | 56,56 | 1024,65535 | 16 |
| R2 | 010* | 001* | 70,80 | 20,20 | 4 |
| R3 | 010* | 01* | 0,65535 | 2688,2688 | 16 |
| R4 | 000* | 10* | 56,80 | 0,65535 | 6 |
| R5 | * | * | 0,6553 | 2688,2688 | 16 |

**Table 2** **An example of converting source prefix addresses to a numerical range.** Each prefix at the second column of the table is converted to corresponding upper and lower boundaries which are presented at the third and fourth columns. The fifth and sixth columns corresponds to decimal representation of the start and end points of the boundary.

| Rule | Source prefix addresses | Lower boundary | Upper boundary | Start | End |
|---|---|---|---|---|---|
| R1 | 100* | 100000 | 100111 | 32 | 39 |
| R2 | 010* | 010000 | 010111 | 16 | 23 |
| R3 | 010* | 010000 | 010111 | 16 | 23 |
| R4 | 000* | 100000 | 101111 | 32 | 47 |
| R5 | * | 000000 | 111111 | 0 | 63 |

2) Skip List: The collection of rules is initially sent to the computer, which calculates the upper and lower bounds of the rules before building skip lists (Pan et al., 2016).

Each of the following fields must be skip-listed: source address, destination address, source port, destination port number, and protocol type. Every skip list comprises a value, a list of pointers to subsequent nodes based on the level of each node, and a list of rules. Each node's level is calculated using a random method that generates protocol skip list which contains nodes with a protocol field equal to their value (0-15 in our implementation), but in other skip lists, lower boundaries of the corresponding fields are less than or equal to the node's value and higher boundaries are more than or equal to it (Chen and Li 1998). The source address, destination address, source port number, and destination port number should be added to the corresponding skip lists in two phases because they have higher and lower bounds (Chen and Li 1998). In the first step, the bottom border is added. The upper and lower boundary node values for each rule are then compared to the lower boundary node value. (Chen and Li 1998). The ID of a rule is added to the list of lower boundary rules when the lower boundary value falls inside the range of that rule. A list of pointers is then constructed for the newly generated node based on the node's level. The top limit is added in the next phase. This action is comparable to inserting the bottom border. The procedure for making a skip list for Table 2's source address field is shown in Figure 2. In fig 2 R1 rule is added to the skip list . First, the skip list is updated to include the lower threshold of 32 at level 0.Because the number 32 lies between R1 and R5, the ID of these rules is added to the rules list. The IDs of the R1 and R5 rules are then added to its rules list, and the higher boundary of 39 at level 2 is placed. The R2 rule has been added to the skip list in Fig. 2B. The IDs of R2, R3, and R5 are added to its rules list together with the value of 16 at level 3. The IDs of R2, R3, and R5 are then added to its rules list together with the value of 23 at level 1. The R3 rule is added to the skip list in Fig. 2C.




G. Packet Categorization

The classification of packets using skip lists and splay trees is as follows. A packet's header contains information that is extracted upon receipt, including the protocol type, source and destination addresses, source and destination port numbers, and other information. Next, a skip list or splay tree is formed for each of the aforementioned fields in the packet, and it is simultaneously searched for a matching node. Any list or tree's search results come with a list of matching rules. An intersection operation is carried out between the five lists acquired from the splay trees in order to identify a common rule between them. The intersection's outcome can be null or include many rules. If the outcome is null, the default rule's associated action is applied to the packet;

otherwise, the highest priority rule's associated action is applied.The rule with the shortest row number has the highest priority because the rules were initially organised according to priority.The preceding section explained splay trees, that are binary search trees that self-adjusting so that the deepest met surviving node in every operation becomes the root following the operation. Splay trees don't keep track of balance or weight, but because they execute numerous tree rotations after each access, they are less usefully efficient than skip lists in many situations. When nodes are enhanced with auxiliary structures, these rotations might be especially destructive. The classification of packets includes this circumstance. When there is reference locality in the process cycle, simple operations like move-to- root might assist to partially overcome this issue and improve the performance of the splay trees. However, it is not ideal in the case of packet categorization where the sequence of the activities has no known locale.(Sahni Kim, 2002).

Instead, compared to splay trees, skip list algorithms are simpler, quicker to construct, and offer notable constant factor speed benefits (Dean Jones, 2007). Their plan is made to operate well as expected for busy access patterns (Sahni Kim, 2002). According to Sen (1991) and Kirschenhofer, Martnez, and Prodinger (1995), great space is saved by skip lists.
To practically investigate the above predictions about the performance of these two competitor algorithms, we imple- ment and experiment them on several data sets.

## II. EVALUATION

On a computer with an Intel Core i5 2.30 GHz processor and 4 GB of RAM, the C++ implementations of the splay tree and skip list techniques were run 10 times each. The performance standards were computed based on average outcomes.

The two methods were compared in terms of memory utilisation, classification time, and the quantity of memory accesses required for packet classification. Rule sets and packet headers were generated using the Class Bench tool (Taylor Turner, 2007). In the assessments For our evaluations, we produced a set of 8 k, 32 k, and 128 k packet headers for every set of rules. The number of memory accesses needed for packet classification as well as the processing times for classification with 500, 1K, and 8K rules were compared using the ACL and IPC rules. To determine how much memory there is In addition, we used ACL and 1000 IPC rules. The time from when a packet first enters. The packet classification time refers to the structure of the classifier and the time at which the system may determine the matching rule for a given packet. The structure of the classifier will operate properly the quicker the packet classification task can be finished. The skip list and splay tree are governed by sets of 1k, 500 and 8k ACL and IPC rules, respectively.

The period of time from the moment a packet enters the classifier's structure and the point at which the system can identify the matching rule for that packet is known as the packet classification time. The classifier's structure will be more effective the faster the packet classification time can be completed. In accordance with sets of 500, 1000, and 8000

ACL and IPC rules for the skip list and splay tree, Figure 3 illustrates the time required to categorise a wide range of packets. These two methods are compared in Figure 3A for 8000 packets.500 rules show the smallest difference between the two approaches in these charts, whereas 8k rules show the biggest difference. The splay tree and skip list classify packets for 500 IPC and ACL rules in 1,011 and 2,271 ms and 391 and 1,415 ms, respectively. Additionally, whereas the splay tree takes 684 and 8,131 ms for packet categorization, the skip list for 8 k IPC and ACL rules takes 805 and 4,231 ms, respectively. We

can draw the conclusion that as the number of rules increases, the classification time disparity between the two techniques' results increases. In actuality, the skip list completes this task better than the splay tree. Additionally, the kind of rules used and their compatibility with IPC rules have a substantial impact on how quickly packets are categorised. A smaller number of rules would result in a smaller time difference, whereas a larger number of rules would result in a larger time difference. Therefore, performance may be impacted by the sort of rules chosen for packet classification.

The skip list and splay tree for 32k packets are both evaluated in Figure 3B. As shown in Fig. 3A, Between ACL and IPC rules, 500 rules demonstrate the least variation in packet categorization time, while 8 k rules show the highest difference. In 1,849 and 4,660 milliseconds, respectively, the skip list classifies packets for 500 IPC and ACL rules, and in 5,981 and 7,994 milliseconds, respectively, the splay tree. Additionally, the packet categorization times for the 8 k IPC and ACL rules skip list are 3,192 and 3,813 milliseconds, respectively.Additionally, the splay tree takes 11,947 and 25,722 milliseconds to categorise packets whereas the 8 k IPC and ACL rules skip list require 3,192 and 3,813 milliseconds, respectively. while the greatest change is shown with 8 k rules. The skip list classifies packets for 500 ACL and IPC rules in 1,849 and 4,660 milliseconds, respectively; the splay tree does it in 5,981 and 7,994 milliseconds, respectively. ACL rules may be used to handle a higher volume of packets if the number of rules is minimal; otherwise, IPC rules should be used for higher rule counts. The distinction is crucial when employing 8 k rules for classification. skip list accomplishes it in 6,792 ms. Similar to the preceding section, the set of 500 ACL rules exhibits the least difference between these two approaches, with the skip list's packet categorization time being 11,846 ms and the splay tree's being 13,020 ms. When using IPC and ACL rules, there is a sizable difference in packet classification time between a skip list and a splay tree. This outcome can be used to choose the best rules for creating a system that is effective in packet classification. gain, the scenario with 8 k rules demonstrates the biggest distinction between the two methods. Using 8 k IPC and ACL rules, the skip list categorises packets in 9,508 and 9,778 milliseconds.splay tree completes the same operation in 22,457 and 72,444 ms. As can be observed, the IPC rules have a significantly lower time difference than the ACL rules.

Generally speaking, Fig. 3 demonstrates that the skip list strategy classifies packets more quickly. Additionally, the skip list's enhanced packet categorization time is substantially lower than the splay tree's due to the higher number of rules than the latter. It is clear that for packet categorization, the skip list plays best than the splay tree. The speed of the search is one of the most crucial factors in determining how well categorization algorithms operate.

The most significant cause of lengthy packet command execution in network processor architecture is memory access. System performance is decreased by frequent memory access. Reduced memory access would speed up the operation by reducing the amount of time needed to classify packets. Reduced memory access is therefore essential to an approach's effectiveness.

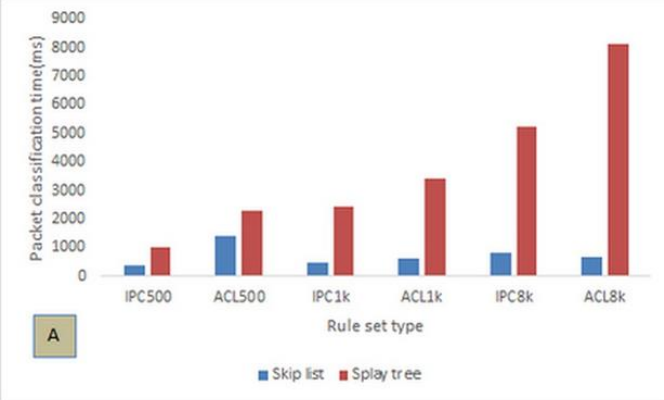
The two methods are compared for 8k packets in Figure 3A. As can be seen, skip list uses less memory than splay tree in every situation. There are 500 IPC rules in the skip list, which has a minimum of 65,477 memory accesses. Splay tree has the maximum number of memory accesses in our evaluation (477,664 with 8 k ACL rules). The difference in memory access between the two systems considerably widens as the number of rules rises. The largest difference in the amount of memory visits made by

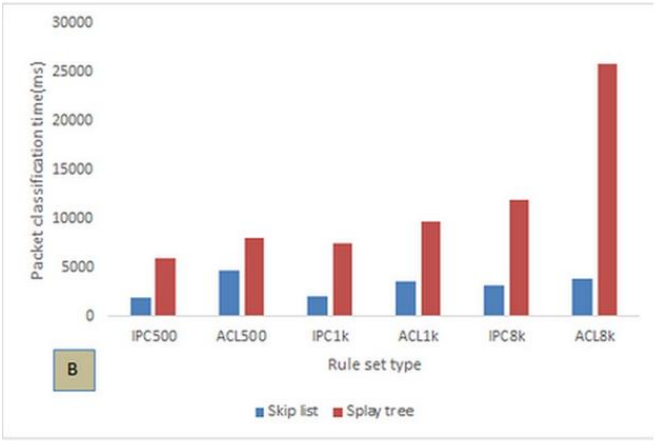
the skip list and splay tree may be seen in the case with 8 k ACL rules. The splay tree has 198,664 and 477,664 memory accesses, compared to the skip list's 104,017 memory accesses and 98,476 memory accesses with 8 k IPC and ACL rules. but the splay tree has 198,664 and 477,664 memory accesses, respectively. The example with 8 k ACL rules shows the biggest disparity in the number of memory accesses made by the skip list and splay tree, with splay tree making 379,188 times more memory accesses than skip list. For 128 k packets, Figure 3B contrasts the skip list and splay tree. The skip list performs better than the splay tree in terms of memory access, as in earlier parts. The minimal number of memory accesses, which generally falls within the skip list with 500 IPC rules, is 215,169. The maximum number, 1890056, is part of the splay tree and has 8 k ACL rules. Time required for packet classification for sets of 500, 1K, and 8K ACL and IPC rules for various packet counts. (A) 8k, (B) (B) both (C) 32k and 128k packets. Memory use for 1K ACL and IPC rules is shown in Fig. 3. The memory consumption of the splay tree and skip list algorithms is shown, respectively, by the red and blue bars. Furthermore, the increase in memory accesses for the skip list with more rules is much less evident than for the splay tree. As a result, it may be said that The skip list excels the splay tree in terms of efficiency. significant finding is the precise agreement in all instances between the results of memory access time and number. The quantity of memory utilisation is another performance requirement for classification algorithms. Therefore, lowering memory use should be the goal of every strategy. Figure 5 displays the memory consumption in bytes for the skip list and splay tree.

The splay tree had 6356259 times more memory accesses than the skip list in the case of 8 k ACL rules, which is the biggest difference between the two methods. In regards to memory access, Figure 4 shows how the skip list surpasses the splay tree. Additionally, the skip list's rise in memory accesses with more rules is far less pronounced than the splay tree's. As a result, it may be said that the skip list performs more effectively than the splay tree. According to the charts in Figures 3 and 4, another significant finding is the precise agreement in all instances between the results of memory access time and number.

Because of memory limits in most systems and the high costs of updating memories, another performance criterion for classification algorithms is memory use. As a result, every approach should aim to reduce memory use. The memory consumption in bytes for the skip list and splay tree used to categorise packets using 1 k ACL and IPC rules is shown in Figure 5. Splay tree requires 30,528 bytes of RAM for IPC rules and 158,440 bytes for ACL rules, whereas skip list uses 31,700 bytes for IPC rules and 162,960 bytes for ACL rules. Skip list consumes significantly more memory than splay tree with both sets of rules.

The pointers in a skip list are stored in this extra space. Additionally, both methods use substantially less RAM when using IPC rules than when using ACL rules.





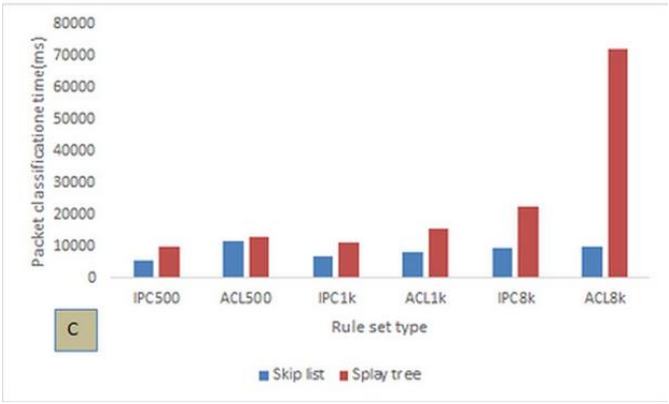
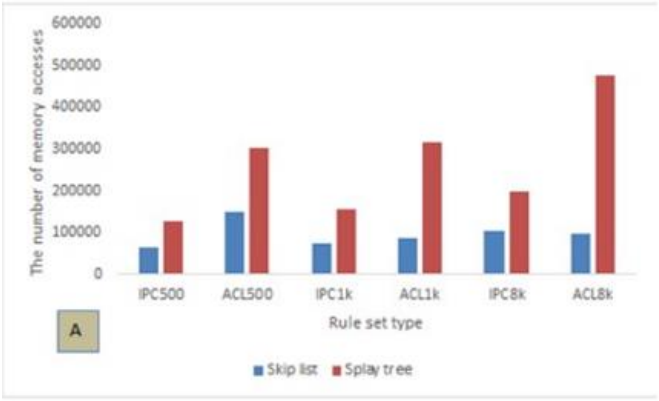
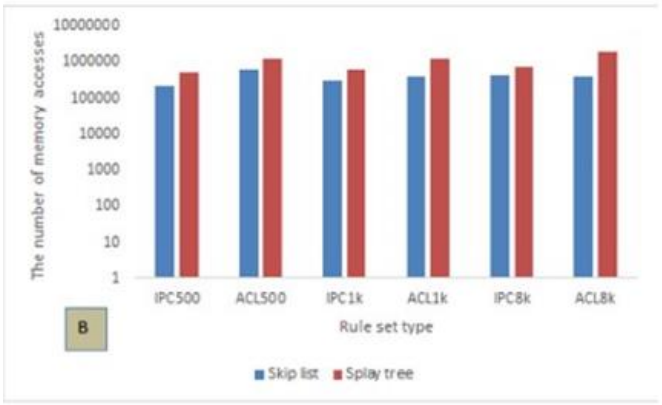




Fig. 1: Packet classification time for the sets of 500, 1000, and 8000 ACL and IPC rules for different numbers of packets. (A) 8k, (B) 32k, and (C) 128k packets.



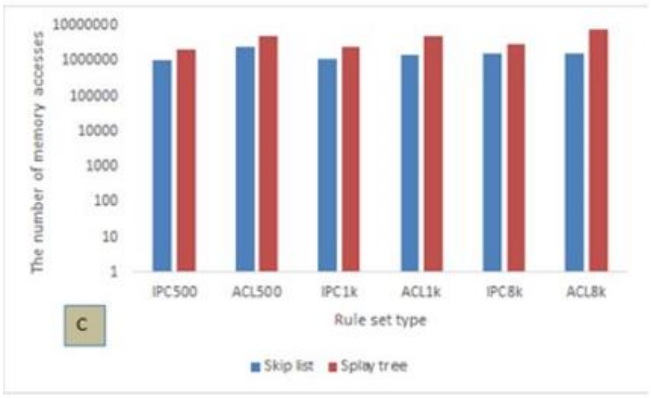





Fig. 3 : Memory usage for 1k ACL and IPC rules. The red and blue bars represent the memory usage of the splay tree and skip list algorithms respectively

## III. CONCLUSION

One of the fundamental operations of network processors is Packet classification. Selecting a packet categorization system that really can catch pace with network speed is the critical issue. The use of memory should be optimized using such a method. Memory and time usage cannot be balanced using the current

techniques. Binary trees, on the other hand, function well when the members are introduced unintentionally but degrade when the operations are carried out sequentially. Because of this, we concentrated on the skip list and the splay tree. Our findings indicate that the skip list performs better in regards to package categorization time and memory visits.

Additionally, when there are more rules, a skip list takes less time and uses less memory than a splay tree to classify packets. Due to the pointers being stored in skip lists, the skip list uses a little bit more memory than the splay tree. The large decrease in memory accesses and packet categorization time in skip lists, however, can be used to appropriately justify this extra space. Therefore, it may be said that the skip list is preferable than the splay tree. The authors want to examine the parallelization of both approaches on graphics processors and assess how well their parallelized versions perform in future research. It goes without saying that the algorithms' data and control dependencies will effect how well they function in parallel processing.

## IV. ACKNOWLEDGMENT

This paper and the research behind it would not have been possible without the exceptional support of our supervisor Prof.Sumit Shevtekar, ME Computer Engg., Pune Institute of Computer Technology. His enthusiasm, knowledge and exacting attention to detail have been an inspiration and kept our work on track from our first encounter of this topic to the final draft of this paper.

## V. REFERENCES

[1]. Efficient Packet Classification using Splay Tree Models Srinivasan, T., Nivedita, M. Mahadevan, V.: Efficient Packet Classification Using Splay Tree Models. IJCSNS International Journal of Computer Science and Network Security 6(5), 28–35 (2020)

[2]. Khezrian N, Abbasi M.2019. Comparison of the performance of skip lists and splay trees in classification of internet packets.PeerJ Computer Science 5:e204

[3]. Analysis of an optimized search algorithm for skip lists. Theoretical Computer Science.2018 by Kirschenhofer, Mart´ınez Prodinger (2018) Kirschenhofer P, Mart´ınez C, Prodinger HJTCS.

[4]. Efficient memristor-based architecture for intrusion detection and high-speed packet classification by V Bontupalli , C Yakopcic, HasanR,TahaTM. 2021. ACM Journal on Emerging Technologies in Computing Systems 14(4):41.

[5]. Fast packet classification algorithm for network-wide forwarding behav- iors by Inoue T, Mano T, Mizutani K, Minato S-I, Akashi O. 2018. : A fast and scalable IP lookup engine for GPU-based softwarerouters by Li Y, Zhang D, Liu AX, Zheng J. 2013 In: The proceedings of the ninth ACM/IEEE conference on networking and communications system designs. Piscataway: IEEE Press.

[6]. A survey and classification of Several different packet classifying techniques by B. Nagpal, N. Singh, N. Chauhan, and R. Murari. 2015.

**Cite this article as :**

Prof. Sumit S Shevtekar, Harish Sumant, Abhijit Suryawanshi, "Splay Tree and Skip List: Effectiveness and Analysis in Internet Packet Categorization", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 6, pp. 285-294, November-December 2022. Available at doi : https://doi.org/10.32628/CSEIT228623
Journal URL : https://ijsrcseit.com/CSEIT228623