Advancements in High-Speed Packet Classification Algorithms with Efficient Rule Set Updates: A Comprehensive Review

Wamane Sangita J
Department of Computer Science & Design
D.V.V.P. College of Engineering
Ahmednagar, Maharashtra
✉ wamanesangita6@gmail.com

Agarkar Balasaheb S
Department of E&TC
SRES's Sanjivani College of Engineering
Kopargaon, Maharashtra
✉ hodetccoe@sanjivani.org.in

ABSTRACT

Essential technology for next-generation network services is packet classification which important network applications like virtual networks, firewalls, Quality of Service (QoS), network security and some more network services. This task involves utilization of predetermined set of rules for categorizing data packets effectively. Assessment parameters such as search speed, memory requirements, filter set size, rapid updates, scalability, flexibility in specification, power consumption, and space requirements are integral to packet classification. Although existing schemes offer high-performance packet classification, they often face substantial performance degradation. Managing multiple fields packet classification poses formidable challenge. Improved packet categorization capabilities are required to facilitate quick rule-set updates and account for dynamic network architecture due to continued expansion of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) in recent years. This paper introduces innovative algorithms designed to achieve high-speed packet classification with the added capability of facilitating fast rule-set updates.

KEYWORDS *: Quality of service, Rule set, Packets, Search speed, Filter set size, Fast updates, Scalability, and power consumption.*

INTRODUCTION

For switches, routers, firewalls, load balancers, and other network appliances to enable security [1], quality of service [2], [3], and other sophisticated services [4], packet classification is one of the most important processes. Packet classification aims to classify packets into discrete "flows," wherein packets within the same flow follow pre-established rules and are processed similarly. These appliances generally have stable rule sets, and fast packet matching and forwarding are dependent on a well-thought-out data structure.

Specific source port numbers are more repeatedly identified than specific destination port numbers in the rule-set databases.

Figure 1 shows an example of a rule that includes many fields. Typical fields include protocol type in the packet header, source port number, destination port number, and IP source and destination prefixes. The research also included a sentence that contains the following points:

Bits 0–4 of the first octet and bits 16–32 of the third and fourth octets are where the bits in the source and destination IP addresses in the rule set are distributed.

1. The rule-set databases more frequently identify precise source port numbers than specified destination port numbers.
2. The size of source and destination port extensions in rule-set databases is typically higher.
3. In the rule-set databases, the number of rules with a single destination port is greater than the number of rules with corresponding source ports [16].

Various methods and structures have been developed over time to create efficient solutions for packet classification. But now that Software-Defined Networking (SDN) and

Network Function Virtualization (NFV) are becoming more popular, many are focusing on software solutions that use product memories, such DRAM or SRAM [5, 6, 7]. The decision tree is an approach that shows assure for obtaining effective packet categorization in the framework of software-based solutions. Though many of them [8], [13], [14], [15],do not support frequent rule-set updates, up to date decision tree- based schemes exhibit extremely good classification performance.

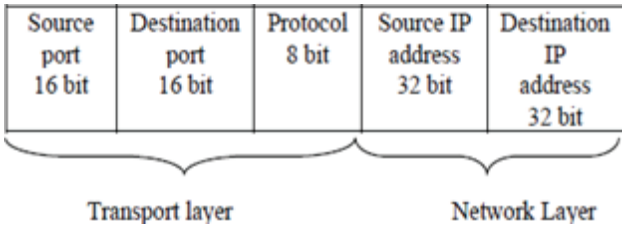

Fig 1. Attributes employed in categorizing packets

PACKET CLASSIFICATION REQUIREMENTS

The functions and the position where classification takes place determine the criteria for packet categorization. The categorization procedure needs to take the least amount of memory and time and possible while operating at line speeds like T1, T3, OC3, OCI2, and OC48 [5]. The following is an outline of the particular requirements:

1. Resource Limitations: When classifying packets using packet classification (PC), factors to be taken into account are the amount of time needed for each packet and the memory use involved. With the availability of different access rate such as T1, T3, OC3, OCI2, and OC48, the PC solution must minimize memory utilization while meeting the performance requirements.
2. Rule Count: The number of rules on a PC can differ depending on the application, like firewalls or backbone routers.
3. Number of information Used: The count of fields within the IP header leveraged for sorting differs across packet classification applications.
4. Character of Rules: Rules can apply general or prefix masks to target IP addresses.
5. Updating Rule Sets: Without sacrificing access performance, the PC system should be able to adjust to adjustments in rules brought about by changes in routes or policies.
6. Worst Case vs. Average Case: Worst-case scenarios should be prioritized over average cases in order to maximize PC performance.

CLASSIFICATION SPECIFICATION

Specification of the Problem

The challenge in Packet Classification (PC) lies in assigning a packet to its correct flow utilizing at least one element from its header, spread over various dimensions or fields. The set of rules, R, delineates ranges for these dimensions, where each rule has an associated cost and defines permissible values for each dimension. The objective is to prepare these rules in advance and, upon receiving a packet characterized by distinct values for each field, identify the rule with the minimum cost that encompasses all of the packet's field values. The complexity of this task can be one-dimensional, two-dimensional, or multi-dimensional, depending on the number of fields considered.

One-Dimensional Classification

In One-Dimensional Classification, the objective is to find the most cost-effective rule that encompasses a query point q within the range [1, U], drawn from n overlapping interval criteria, each with its respective cost. This type of classification includes distinct sub-problems: IP lookup (IPL), which targets matching queries to IP addresses, and Range Location (RL), which handles non-overlapping intervals spanning the entire range [1, U].

Two-Dimensional Classification

For two-dimensional classification, the setup includes n rectangles on a grid ranging from [1, ..., U] by [1, ..., U], each assigned a specific cost. The aim is to set up a system that allows for the rapid determination of the least expensive rectangle from R that covers a given point q on the grid.

Multidimensional Classification

In multidimensional classification, the goal is to find the most economical rectangle within set R that includes a d-dimensional point q.

CLASSIFICATION ALGORITHMS

Prior studies on packet classification have concentrated on achieving optimal performance, whereas this research aims to enhance packet classifiers within SDN and NFV contexts, emphasizing both superior classification and update performance. The focus lies on expediting rule-set modifications, with literature categorized into decision tree-based, tuple space-based, and hybrid schemes.

Scheme based on decision trees

Decision trees, a prominent method in packet classification, arrange rules into a tree-like structure, beginning with all rules located at the root node and dividing them into subspaces using methods such as equal-sized cutting or equal-dense splitting, until reaching leaf nodes with fewer than a specified threshold (binth). Traversing the tree during packet classification is succeeded by a linear search within the leaf node to identify the best rule match.

Introduced by HiCuts[9], the notion of partitioning the search space through equal-sized cutting has advanced with HyperCuts[14], featuring a more adaptable cutting strategy utilizing multiple fields per phase and memory-saving improvements. Nonetheless, both HiCuts and HyperCuts face challenges like rule repetition and excessive memory usage attributed to equal-sized cutting.

Table 1 presents a collection of 2-D rules, with R1 identified as the most critical among the five rules listed. Refer to Figure 1 for further details.

HyperSplit[12] introduces an equal-density splitting technique, dividing a search area into two sub-spaces with different rule counts to tackle rule replication, providing greater flexibility in selecting the splitting line and avoiding rule scattering across sub-spaces. Despite its effectiveness in reducing rule replication compared to cutting-based trees, HyperSplit often results in taller tree heights and could lead to exponential memory usage with larger rule-sets.

EffiCuts[13] diverges from HyperSplit by addressing rule replication through partitioning the rule set into subsets and forming a HyperCuts decision tree for each subset, leading to notable reductions in memory usage without significant performance drawbacks. HybridCuts[12], an extension of EffiCuts, presents a fresh partitioning technique that yields fewer partitions. SmartSplit[15] improves performance by forecasting rule-set traits and crafting unique basic decision trees for various subgroups. NeuroCuts[8], a recent development, asserts superior classification performance and diminished memory usage compared to heuristic-based methods through the implementation of deep reinforcement learning for decision tree construction.

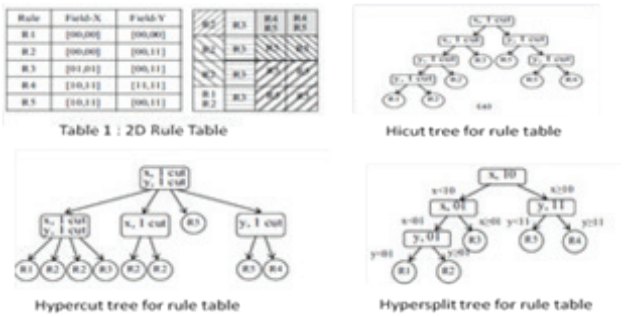

Fig. 2. HiCut, Hypercut and Hypersplit trees for rule set

CutSplit [1] and PartitionSort [6] assert rapid rule-set adjustments. Yet, CutSplit's update times can significantly exceed those of tuple space methods, sometimes reaching several milliseconds in specific cases. PartitionSort recommends dividing the rule set into sortable subsets and building a multi-dimensional interval tree for each subset. Despite enabling quick rule-set updates, PartitionSort's classification performance lags behind cutting-edge decision tree schemes due to the greater number of partitions compared to approaches like EffiCuts.

Packet Classification Using Tuple Spaces

Tuple space-based schemes, although less prevalent than decision tree schemes, arrange rules into hash tables based on basic rule attributes, facilitating rapid insertion and deletion with an average of one-memory access for quicker updates. Upon packet arrival, these separate hash tables are queried independently to identify the optimal match.

Classical Tuple Space Schemes

Tuple Space Search (TSS) [9], a core tuple space-based technique for packet classification, dissects classification queries into exact match searches

within hash tables. TSS utilizes pre-calculated tuples to organize rules into distinct hash tables, where concatenated unique bits from each field compose a tuple to map rules to the corresponding hash table via a hash key. For example Since Table II rules R1 and R2 utilize three and zero bits in their corresponding two fields, they would share a tuple space, as illustrated in Table III, demonstrating how TSS forms four tuple spaces for the specified rules in Table II. Pruned Tuple Space Search (PTSS) [9] enhances TSS by selecting a subset of potential tuple spaces through individual analysis of each rule field. However, the surplus of tuple spaces diminishes classification speed for both PTSS and TSS, necessitating scanning of every tuple space for each packet, especially challenging in classifiers with numerous fields such as OpenFlow classifiers.

Rule id	Priority	Field X	Field Y	*Action*
R_1	6	111*	*	$action_1$
R_2	5	110*	*	$action_2$
R_3	4	*	010*	$action_3$
R_4	3	*	011*	$action_4$
R_5	2	01**	10**	$action_5$
R_6	1	*	*	$action_6$

Table 2. An Illustration of 2-Tuple Classifier

Tuple	*Rule id*	Rule Priority	Tuple Priority	Field X	Field Y	*Action*
(3, 0)	R_1	6	6	111*	*	$action_1$
	R_2	5		110*	*	$action_2$
(0, 3)	R_3	4	4	*	010*	$action_3$
	R_4	3		*	011*	$action_4$
(2, 2)	R_5	2	2	01**	10**	$action_5$
(0, 0)	R_6	1	1	*	*	$action_6$

Table 3. TSS Generate 4 Tuples for Rules Outlined in Table 2

Recently Introduced Tuple Space Schemes

One such scheme, called TupleMerge [7], loosens the constraints on rule placement inside the same tuple space, which improves TSS. TupleMerge shortens the total time required for classification by reducing the number of candidate tuple spaces by combining tuple spaces that have rules that share comparable attributes. But more tuple spaces combined could mean more hash collisions, which could hurt TupleMerge's speed. When used in Open vSwitch, Priority Sorting Tuple Space Search (PSTSS) [7] improves TSS performance by sorting tuple spaces according to each tuple space's pre-computed priority (i.e., the Tuple Priority column in Table III). The search can stop as soon as a match is discovered since it searches tuple spaces in descending priority order, giving it the highest priority among all possible matched rules. While PSTSS may outperform TSS on average, it does not outperform TSS on a worst-case basis.

Hybrid Schemes

Two recent hybrid approaches, CutTSS [15] and CMT [10], aim to balance classification performance and update efficiency. CutTSS employs a strategy where the rule-set is initially partitioned based on small and big fields. CutTSS builds a pre-cutting decision tree by segmenting rules along small fields for the subset with small rules, and uses TSS for packet categorization for the subset with big rules. After that, TSS is used to categorize rules in leaf nodes that have more rules than binth rules. Although CutTSS exhibits enhanced classification performance due to the initial cutting step, it may face challenges with severely imbalanced rule-set distributions, leading to suboptimal performance in certain cases.

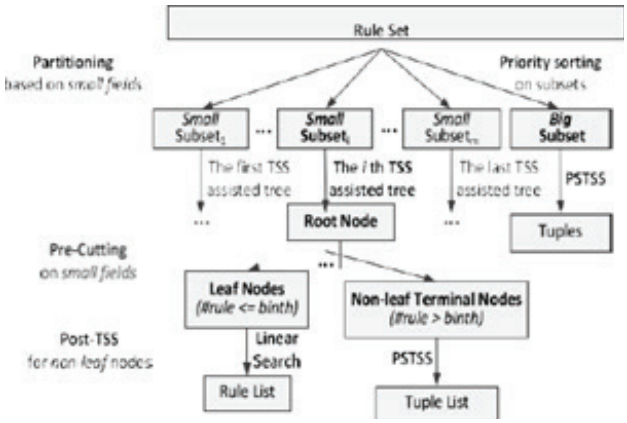

Fig. 3.Refined CutTSS framework

Contrarily, CMT [10] utilizes a common mask tree, connecting each node to a common mask for packet classification. By portraying this shared mask tree as a configuration of hash tables, CMT asserts rapid rule-set modifications and swift packet classification. Nonetheless, CMT's performance diminishes due to its reliance on hashing for rule distribution among sub-spaces, neglecting distribution efficiency and causing a tall tree height. Furthermore, the iterative tree construction until all rules align with the same mask

contributes to an increased tree height and a notable rise in the number of leaf nodes..

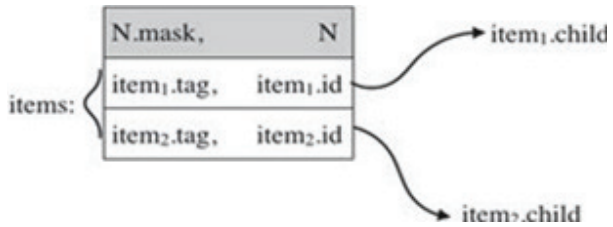

Fig. 4. The node structure in CMT

Table 4. A prefix-set P example

Name	Prefix
p_1	0101
p_2	111*
p_3	1***
p_4	01**
p_5	101*

Fig. 5: The CMT of Table 2's prefix-set P.

As a result, as compared to other sophisticated decision tree schemes or tuple space schemes, CMT shows reduced memory efficiency. Notably, a recently proposed hybrid system called HybridTSS [15] has surfaced, offering a recursive building of tuple levels for packet categorization that is comparable to the strategies that have been addressed..

THE NECESSITY OF CLASSIFYING PACKETS

The performance of internet traffic can be greatly enhanced by packet classification. Services that need to be able to separate and isolate traffic in various flows for proper processing, firewalls, and service quality are among those that depend on packet categorization. Some techniques eliminate floating point division after the packet is categorized. Regarding the query data, the rule match is confirmed. Furthermore, the resources and power are estimated. Network system technological innovations: One intriguing option for a networking system building component is the Network Processor Unit (NPU), which has shown great promise.

By venturing into multi-core network processors, NPU aims to harness the power of parallel processing, allowing for simultaneous execution of multiple tasks and significantly boosting overall performance. Additionally, the exploration of thread-level parallelism further signifies the company's intent to optimize processing capabilities by efficiently managing and executing multiple threads concurrently. In addition, they provide us highly integrated resources and computational capability never before seen. Therefore, in order to release the latest hardware and software technologies from their bottleneck and make them widely available to customers, new packet categorization solutions must be well-suited. When it comes to packet processing, the NPU shows that it can provide a complete solution that includes both forwarding and categorization [12], [13].Network applications are becoming more and more complex: The Internet's expansion and diversity are placing more and more demands on network infrastructure's functionality and performance. Firewalls often filter out unwanted traffic using conventional packet classification algorithms. As contemporary networking devices increasingly incorporate a myriad of network applications, packet classification has become a prevalent method applied across various applications. These include but are not limited to service-aware routing, intrusion prevention, and traffic shaping.

PERFORMANCE METRICS OF PACKET CLASSIFICATION

The evaluation of classification algorithm performance can be assessed based on the following criteria:

- Search speed — Enhancing search speed is imperative for achieving faster classification, especially in the context of high-speed links. For instance, links operating at 10Gbps can potentially handle 31.25 million packets per second, considering the assumption of minimum-sized 40-byte TCP/IP packets.
- Ability to handle large real-life classifiers.

- Low storage requirements —To enable the use of fast memory technologies like SRAM (Static Random Access Memory). SRAM there is requirement of small storage algorithm.
- Fast updates — The need for updating the data structure arises as the classifier undergoes changes.
- Flexibility in specification — A classification algorithm should support general rules, including, operators (range, less than, greater than, equal to, etc.), prefixes and wildcards.

CONCLUSION

Packet classification algorithms are vital in network management, effectively organizing and directing network traffic according to predefined rules. Facilitating rapid rule set updates is crucial for adapting to dynamic network environments characterized by rapidly changing policies, security measures, and traffic patterns. This research introduces an innovative hybrid packet classification method, amalgamating tuple space-based and decision tree-based schemes, to counter performance deterioration during rapid rule-set updates. This approach provides flexibility and responsiveness to changes in the rule set without sacrificing speed, guaranteeing network resilience, compliance, and resource optimization.

REFERENCES

1. W. Li, X. Li, H. Li, and G. Xie, "CutSplit: A decision-tree combining cutting and splitting for scalable packet classification," in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Apr. 2018, pp. 2645–2653.
2. V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun classification," in Proc. 6th Asia–Pacific Workshop Netw., 2022, pp.
3. H. Lim and S. Y. Kim, "Tuple pruning using Bloom filters for packet classification," IEEE Micro, vol. 30, no. 3, pp. 48–59, May/Jun. 2010. (SIGCOMM), 1999, pp. 135–146.
4.. Y. Qi, L. Xu, B. Yang, Y. Xue, and J. Li, "Packet classification algorithms: From theory to practice," in Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM), Apr. 2009, pp. 648–656.
5.. J. Zhong and S. Chen, "Efficient multi-category packet classification using TCAM," Comput. Commun., vol. 169, pp. 1–10, Mar. 2021.
6.. K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algo- rithms for advanced packet classification with ternary CAMs," ACM SIGCOMM Comput. Commun. Rev., vol. 35, no. 4, pp. 193–204, Oct. 2005.
7. S. Yingchareonthawornchai, J. Daly, A. X. Liu, and E. Torng, "A sorted-partitioning approach to fast and scalable dynamic packet classification," IEEE/ACM Trans. Netw., vol. 26, no. 4, pp. 1907–1920, Aug. 2018.
8. J. Daly et al., "TupleMerge: Fast software packet processing for online packet classification," IEEE/ ACM Trans. Netw., vol. 27, no. 4, pp. 1417–1431, Aug. 2019.
9. E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classifica- tion," in Proc. ACM Special Interest Group Data Commun., 2019, pp. 256–269.
10. W. Li et al., "Tuple space assisted packet classification with high performance on both search and update," IEEE J. Sel. Areas Commun., vol. 38, no. 7, pp. 1555–1569, Jul. 2020.
11. S. Chen, J. Zhong, T. Huang, Z. Wei, and S. Zhao, "CMT: An efficient algorithm for scalable packet classification," Comput. J., vol. 64, no. 6, pp. 941–959,
12. Jun. 2021. P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in Hot Interconnects, vol. 40. Los Alamitos, CA, USA: IEEE Computer Society, 1999.
13. B. Vamanan, G. Voskuilen, and T. N. Vijaykumar, "EffiCuts: Optimizing packet classification for memory and throughput," ACM SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, pp. 207–218, Aug. 2010.
14. W. Li and X. Li, "HybridCuts: A scheme combining decomposition and cutting for packet classification," in Proc. IEEE 21st Annu. Symp. High-Perform. Interconnects, Aug. 2013, pp. 41–48.
15. Y. Liu et al., "HybridTSS: A recursive scheme combining coarse- and fine—Grained tuples for packet classification," in Proc. 6th Asia–Pacific Workshop Netw., 2022, pp.
16. H. Lim and S. Y. Kim, "Tuple pruning using Bloom filters for packet classification," IEEE Micro, vol. 30, no. 3, pp. 48–59, May/Jun. 2010.