Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 12, No. 2, (2021),
ISSN : **1906-9685**

BREAKING BARRIERS: ENHANCING SEARCH PERFORMANCE WITH AN IDEAL PACKET CLASSIFICATION ALGORITHM

[#1]**Mrs.BHEERAM SANKEERTHANA,** *Assistant Professor*
[#2]**Mr.JANGA RAVI CHANDER,** *Assistant Professor*
Department of Computer Science and Engineering,
SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES, KARIMNAGAR, TS.

ABSTRACT-
Several methods were explored in an effort to efficiently categorize packets according to their accessibility. As part of our research, we examined several decision tree implementations, each of which was used to classify packets into one of several categories. It was the proximity to an ideal choice that was used to determine the decision tree's initial field and number of cuts. This increased the amount of time and storage space required for the search. To determine the optimal pair and optimal multi-pair for each packet, algorithms exist that make use of decision trees.
With the introduction of cutting-edge network applications, multi-match packet classification has
taken on greater significance. In this situation, in addition to the most crucial rule, you should return the entire set of matched results. Finding effective classification methods is crucial for addressing the challenges associated with this process. We've developed a new method, called "boundary cutting," that efficiently sorts packets. Important matching is required by most conventional applications that categorize packets.

There are two main benefits to the demonstrated
approach. The proposed algorithm improves upon prior methods by using rule boundaries

rather than fixed gaps in its border-cutting functionality. This has significant implications for the required amount of memory. Cutting boundaries makes it more difficult for internal nodes to process information, but binary search is superior at it.

Keywords: Decision tree algorithms, Packet classification, Boundary cutting, Priority matching, Binary search**.**

1. INTRODUCTION:

Servers connected to the Internet have an advantage in providing certain services due to their ability to categorize data packets. Since there is a growing demand for worm detection and network intrusion detection systems, multimatch classification has attracted the attention of many researchers. A chip with a high bandwidth but low on-chip memory is ideal for storing the rule library for packet classification. This will increase the chip's off-chip memory at the expense of performance.
The memory requirements for creating a packet sorting table are critical information to have. The effectiveness of packet classification algorithms is primarily evaluated by how quickly they process data. This is due to the fact that at wire speed, each incoming packet must be categorized. In order to determine the type of packet, the chip must make a number of memory calls outside of the chip.
Our primary objective is to explore different ways of using decision trees to classify packets into distinct categories. Like in Hi Cuts and Hyper Cuts, the initial decision trees' field and number of cuts were chosen with the best possible match in mind. This increased the amount of time and storage space required for the search. In order for this method to be

effective, we must first process it using intricate heuristics that are associated with each set of rules. Since the rule set is kept off-chip and the internal nodes of the tree are kept on-chip, the decision tree method can search very quickly.

Using a decision tree, it is clear which packets need to be prioritized for sorting based on which matches are most critical. Newer network applications necessitate a method for categorizing packets into multiple subsets, while simultaneously displaying all matching results and the most crucial matching rule. Problems with categorizing data require robust and capable computer programs to be developed as a solution. In our research, we developed a novel boundary-cutting-based method for clustering packets. By strategically placing each rule, it ensures that each rule can efficiently remove the required amount of space. This article demonstrates a novel method for rapidly sorting packets into groups via a slicing operation. By analyzing the borders of the room, the system determines the precise region covered by each rule. As a result, a foreseen method of cutting is more precise than intricate algorithms. It also uses less memory and operates quicker.

The number of rules that apply to a subspace cannot be reduced by partitioning due to the fact that Hi Cuts and Hyper Cuts operate on a constant interval. The boundary cutting (BC) algorithm is demonstrated in our research. It's a tried-and-true method for making clean cuts at the edge of each rule. When dividing a prefix plane along rule boundaries, both the beginning and ending edges of each rule are considered. However, it doesn't matter which way you go about it; all decision tree methods care about is which subspace an input packet resides in. In the first step, all of the rule fields that correspond to the subspace identified by the headers of the input data are matched. Different cuts are made at various times at each internal node of the border cutting decision tree. In order to determine which branch to take given an input, a binary search must be performed at each node in the tree.

2. OVERVIEW OF EARLIER DECISION TREE AL-GORITHMS:

Assuming k = 1, rule Rk should be applied to packet P. All header fields must be set to Fd if d = 1, yielding a False result. In Rk, how many fields are available for use? Assuming N rules and D fields, we get the following expression: (d) N = D * N. Only the most crucial rule is returned when multiple rules are equally applicable to a packet classification problem. It is possible to use multiple matching rules when sorting packets into categories.

Most sets of rules are broken down into five sections. The first two fields, which concern the source and destination prefixes, need to be matched. We'll dive deeper into the topic of sending and receiving ports that are compatible in the next two sections. The number of cuts required for a given field is determined by a comparison to the protocol-dependent constant field.

The binth is a set of rules used to configure the leaf nodes of a decision tree. The rules for conducting a linear search are as follows. Exactly what is a regulation, then? A rule is the region in a two-dimensional (2D) plane containing the first two prefix fields. which regions of the prefix plane are covered by each rule in a given 2-dimensional rule set. The rule's longest field length (W in IPv4) is displayed, and the corresponding space is included in the rule.

3. EXISTING SYSTEM:

Our primary objective was to explore various applications of decision trees for logically clustering data packets. If the decision tree process is broken up into two stages, it can be completed much more quickly. On-chip memory is used for storing the internal tree nodes, while off-chip memory houses a massive library of rules. In addition to identifying the most crucial match, choice tree algorithms can determine the most effective way to cluster multiple packet matches. Traditional decision tree methods like Hi Cuts and Hyper Cuts used a best-in-class decision to determine the field and number of cuts. Both the search time and storage requirements are reduced. There is preparation required before attempting to use this technique. This necessitates the development of intricate heuristics for each distinct set of rules.

4. DISADVANTAGES:

Assembling the pre-processing calculations takes a lot of time and resources.

Building decision trees is a memory-intensive process. Algorithms will be expanded to

accommodate more rules sets as a workaround. The rules are cut at a predetermined time regardless of the amount of land each rule exceeds. This method is therefore obsolete.

5. PROPOSED SYSTEM:

- This paper demonstrates a novel and practical method for organizing these packets by cutting their perimeters. The number of rules is reduced by the proposed method because each rule addresses a different topic.
- The packet classification table of the proposed algorithm can be predicted with reasonable accuracy. That is to say, unlike previous decision tree methods, it does not require the use of complex heuristics

6. ADVANTAGES OF PROPOSED SYSTEM:

- The proposed algorithm improves upon prior methods by using rule boundaries rather than set boundaries to perform border cutting. Less RAM is required as a result.
- Binary search is effective for searching at internal nodes, which BC cannot index.

7. IMPLEMENTATION:

Building a BC Decision Tree:

Rules' beginning and ending points provide natural divisions in a prefix plane. However, it makes no difference which you pick because decision tree methods typically seek out a subspace that an input packet is in. Decision tree leaves are used to compare input file headers against rules for a specific subspace.

Searching in the Boundary Cutting:

The BC decision tree has no predetermined time intervals between cuts at any given node. In order to locate the correct edge for each key, a binary search must be performed at each node within the tree. The pointer to the daughter node is retained if the value you enter is greater than or equal to the value you are searching for. This can be seen by analyzing an incoming packet with the following headers: (000110, 111100, 19, 23, TCP). A binary search is performed on the input's header at the root node.

Header 000110 is compared to the middle entry 010000 in the parent node. Since the size of the input has been reduced, the search is constrained to a narrower region, and only the input and the entry 000100 are examined. Since the input has expanded, the child indicator on the second edge is recalled, and the search is conducted among a larger set of candidates. The input appears smaller when compared to the value 001000, but the smaller value does not contain any records. Since a signal was located, the search has shifted its attention to the second edge. The binary search algorithm chooses the second-to-last edge for the 111100 header. Rules in the leaf node are discovered using linear search, just as they are with HiCuts and HyperCuts.

Selective Boundary Cutting:

It improves the efficiency with which the BC algorithm can be used in this application. Decision trees, including the BC algorithm, use binth to determine whether or not a given area is a leaf node. An "internal node" is a node that has more rules than binth, hence the name. If so, we call it a leaf node. The BC algorithm selects a subspace as an internal node, and the cutting lines within that subspace serve as the leading edges of the rules being applied. Let's discuss an innovative application of the binth for modifying or erasing the rule's limit at an internal node. When the number of rules in a partition reaches the binth level, the better framework prevents any of those rules from crossing over to the other partition.

Data Structure:

There are two ways to store rule sets in decision trees. The first approach contrasts rule tables with decision trees. Each rule will only be stored in the rule table once. But in a decision tree, each leaf node is connected to the rule table that governs it. Each leaf on the binth has a corresponding rule point requirement, which is listed here. When searching for the optimal rule for a packet or the complete set of matched rules, more memory accesses are required to retrieve the rule table. This is so because you can easily count the number of rules in a node's leaves. The second method employs leaf nodes as storage locations for rules.

This reduces the time it takes for the search to complete because fewer rule tables need to be consulted. However, because the rules are being used twice, significantly more memory is being consumed. It is more important to consider search speed than memory limits in this simulation if you want to keep rules on leaf

nodes.

8. RELATED WORK:

The concept of packet classification is fundamental to many Internet services, including traffic monitoring and firewall packet filtering. Businesses typically use ternary content addressable memories (TCAMs) to partition fast packets into subsets. TCAMs simultaneously check each message against all three rules and sort them in real time. Because packet categorization rules frequently refer to fields as ranges, converting them to TCAM-compatible rules can result in a large number of rules, a problem known as "range expansion." If TCAMs have plenty of space for data, then this shouldn't be an issue. TCAMs, alas, have a very limited capacity for energy storage. Additionally, the stricter the rules are, the more energy and heat are generated.

Year after year, new services are introduced to the Internet, and new rules are added to packet analyzers. This research seeks to answer the question of how to create a packet classifier with the same semantics as an existing one while using as few TCAM records as possible. The TCAM Razor is discussed in this research. It's a great boon, resource, and time saver. TCAMs, or thermal control and monitoring devices, are extremely pricey. TCAMs have a cost that is 30 times higher than that of double-data-rate SRAMs for each stored bit. On the other hand, for a

You may need 2(L-1) TCAM entries if your port range field is only bits long. This necessitates exploring alternative algorithmic approaches. In th

is case, k = 1 indicates that rule Rk matches packet P. Packet classification describes this process. If the value of d is 1, and every header field is Fd, then the result is False. If we have N rules and D fields, then d will tell us how many fields are in Rk. When there is only one rule that fits a given packet classification problem, the most relevant rule is returned. However, in a packet classification problem, more than one rule may apply. In that case, a set of all applicable rules is provided.

Five categories are typical for rule sets. Source and destination prefixes in the first two fields must be matched. We'll dive deeper into the topic of sending and receiving ports that are compatible in the next two sections. To determine the number of cuts required for the target variable, a perfect match is required for the protocol-dependent, unchanging variable. The binth is a set of rules used to configure the leaf nodes of a decision tree. The rules for conducting a linear search are as follows.

HiCuts *:*

Each rule generates a five-dimensional hypercube in the given space. The header of each data packet specifies a location within that region. It employs a back-and-forth procedure to segment the space into subspaces along each dimension individually. The smaller number of intersecting rule hypercubes in each subspace makes this possible. Increasing the number of cuts in HiCuts allows you to incorporate a greater depth of information into your decision tree. In contrast, clearing out some cuts slows down the investigation. Finding a happy medium between the need for storage and the need for fast search is challenging. To optimize the heuristics, the HiCuts technique adjusts two parameters: a space factor (spfac) and a threshold (binth). Both the amount of memory used and the depth of the decision tree are controlled by these variables.

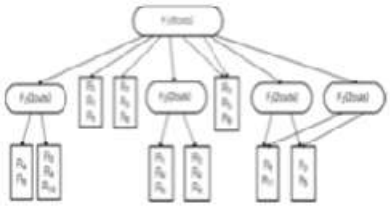

Fig:-HiCuts algorithm.

Exactly what is a regulation, then? A rule is the region in a two-dimensional (2D) plane containing the first two prefix fields. Each rule in the provided 2-dimensional example set addresses the prefix plane. Where W is the largest field length (32 in IPv4), F2, and field lengths (i,j), the rule applies to an area of 2(w-i) * 2(w-j).

Hyper Cuts:

When deciding on cut dimensions, the Hyper Cuts algorithm considers many fields simultaneously, while the HiCuts approach considers only a single field at a time. The algorithmic decision tree generated by the Hyper Cut method using the same data. The value of the binth is set at 1.5, and the value of the space is set at 3. In this case, "and" is used in conjunction with "fields" to make a deep cut. Bitwise combinations of 00, 10, 01, and 11

form the edges of the root node. One bit from the first field and one bit from the second field make up each possible combination.

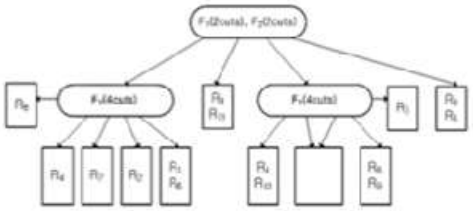

Fig:-hyper cuts

9.DECISION TREE CHARACTERISTICS:

In the second part, we constructed decision trees with a lower bound of binth using the BC, SBC, HiCuts, and Hyper Cuts algorithms. Cutting terminates when the number of rules in a subspace falls below the minimum threshold binth. The effectiveness of algorithms like HiCuts and Hyper Cuts depends, in part, on how much room there is to work with. When a decision tree isn't optimized, the leaf nodes only contain binth rules. Rules that apply to all children of a given subtree are identified by working our way up the tree from the roots. Then, the subtree's root receives these rules. Repeating this procedure until the initial node of the decision tree is reached is the norm. Here, the binth value ensures that the cumulative set of rules observed at intermediate nodes along the path from the root to a leaf node is always preserved. The search speed is barely affected by the optimization, but there are significantly more instances of duplicate rules.\

	Entry width	No. of bits	Field
Boundary Cutting	Node (3.625 bytes)	3	field selection
		5	no. of entries
	Entry (7.75 bytes)	32	boundary value
			child pointer
Selective Boundary Cutting	Node (3.5 bytes)	3	field selection
		5	no. of entries
	Entry (7.625 bytes)	32	boundary value
			child pointer
HiCuts	72.375 Bytes	3	field selection
		9	number of cuts (512 cuts max
		512	child map
			pointer to the first internal chi
			pointer to the first leaf child
HyperCuts	72 Bytes	3	field selection
		13	number of cuts in multiple fie
		512	child map
			pointer to the first internal chi
			pointer to the first leaf child

Fig:-DATA STRUCTURES OF INTERNAL NODES IN EACH DECISION TREE

Rule	N	Src Prefix No.	Src Prefix Rate(%)	Dst Prefix No.	Dst Prefix Rate(%)	Src Port No.	Src Port Rate(%)	N
ACL1K	958	3	0.31	1	0.10	958	100	2
ACL5K	4660	8	0.17	11	0.24	4660	100	14
ACL10K	9735	19	0.20	30	0.31	9735	100	30
ACL50K	48420	35	0.07	122	0.25	48420	100	98
ACL100K	95340	60	0.06	229	0.24	95340	100	168
IPC1K	938	27	2.73	12	1.21	835	84.31	5
IPC5K	4460	76	1.70	75	1.68	3615	80.91	24
IPC10K	8112	148	1.82	124	1.53	6373	78.56	42
IPC50K	49047	184	0.38	191	0.39	36344	74.10	235
IPC100K	94570	371	0.39	315	0.33	68292	72.37	435
FW1K	871	32	3.67	132	15.15	347	39.84	8
FW5K	3067	114	3.72	339	11.05	1167	38.05	30
FW10K	4351	78	1.79	491	11.28	1527	35.10	43
FW50K	49163	1700	3.46	1139	2.32	36788	74.81	147
FW100K	82473	2277	2.76	1564	1.90	61546	74.63	235

Fig:-CHARACTERISTICS OF RULE SETS IN THE NUMBER AND RATE OF WILDCARDS

10. CONCLUSIONS:

The boundary cutting algorithm and clustering are discussed in this article. The dataset is divided into subsets during testing based on their degree of similarity to one another.

These groups are then used to determine whether or not a given packet poses a threat. Since it cuts along the edge of the space, the boundary cutting algorithm is more precise than clustering.

REFERENCES:

1. H. J. Chao, “Next generation routers,” Proc. IEEE, vol. 90, no. 9, pp.1518–1588, Sep. 2002.
2. A. X. Liu, C.R.Meiners, andE.Torng, “TCAMrazor: A systematic approach towards minimizing packet classi- fiers in TCAMs,” IEEE/ACM Trans. Netw., vol. 18, no. 2, pp. 490–500, Apr. 2010.
3. C. R. Meiners, A. X. Liu, and E. Torng, “Topological transformation approaches to TCAM-based packet classification,” IEEE/ACM Trans. Netw., vol. 19, no. 1, pp. 237–250, Feb. 2011.
4. F. Yu and T. V. Lakshnam, “Efficient multimatch packet classification and lookup with TCAM,” IEEE Mi- cro, vol. 25, no. 1, pp. 50–59, Jan. –Feb. 2005.
5. F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz, “Efficient multimatch packet classification for network security applications,” IEEE J. Sel. Areas Com- mun., vol. 24, no. 10, pp. 1805–1816, Oct. 2006.
6. H. Yu and R. Mahapatra, “A memory-efficient hash- ing by multi-predicate bloom filters for packet classifi- cation,” in Proc. IEEE INFOCOM, 2008, pp. 2467–2475.
7. H. Song and J. W. Lockwood, “Efficient packet clas- sification for network intrusion detection using FPGA,”in Proc. ACM SIGDA FPGA, 2005, pp. 238–245.
8. P. Gupta and N. Mckeown, “Classification using hi- erarchical intelligent cuttings,” IEEE Micro, vol. 20, no. 1, pp. 34–41, Jan.–Feb. 2000.
9. S. Singh, F. Baboescu, G. Varghese, and J. Wang, “Packet classification using multidimensional cutting,” in Proc. SIGCOMM, 2003, pp. 213–224.
10. P. Gupta and N. Mckeown, “Algorithms for pack- et classification,” IEEE Netw., vol. 15, no. 2, pp. 24–32, Mar.–Apr. 2001