Comparative evaluation of Recursive Dimensional Cutting Packet Classification, DimCut, with Analysis

Hediyeh AmirJahanshahi Sistani[1], Haridas Acharya[2]
[1,2]Department of Computer Studies and Research, Symbiosis International University, Pune, India
hediehamirjahanshahi@yahoo.com
[2]Allana Institute of Management Science, Pune University, Pune, India
haridas.undri@gmail.com

Abstract— an infinitely expanding number of network appliances are utilising packet classifiers to fulfil Quality of Service, security, and traffic engineering tasks. Packet classification is an important role of firewalls and routers. Internet firewalls, routers an d service providers perform different operations at different flows. All the packets have to be classified methodically for emerging broadband internet services, and applications such as Internet TV/Radio, gaming, Video on Demand (VoD) and e-businesses, which are in perpetual demand for a higher degree of transmission bandwidth, complex security, and specific Quality of Service (QoS).

This paper presents a comparative evaluation packet classification of algorithms, HiCut, based on a decision tree structure, and the Recursive Dimensional Cutting (DimCut). The comparison has been conducted on operations based on similar principles and design choices. Performances measurements have been obtained by placing the implemented classifiers in the same test conditions scenario. In particular, the comparison aims at achieving a good arrangement between performance, memory usage and flexibility.

The Recursive Dimensional Cutting (DimCut) is the extension of HiCut algorithm with new heuristics ideas, implementing techniques and parallel programming features which classify packets while retaining HiCut's basic framework. After testing the DimCut algorithm, for classifying packets based on five header fields, it is observed that the algorithm can classify packets rapidly. Th e DimCut algorithm has two separated levels, pre-processing level (tree construction and making index table) and search level.

Keywords— Firewalls; Rules; Packet Classification; Partitioning; Evaluation; DimCut

INTRODUCTION

Internet traffic comprises of an ordered sequence of packets, where every packet consists of header and data. The data in each packet contains a fraction of the large set of data which has been branched into packets at the source, that later get assembled into the original data when reaching the destination. The main function of a firewall is to inspect and decide about the network traffic in accordance with the security policy. The security policy system clarifies how to process the network traffic. Normally, the security policy and rules are entered manually by a system administrator to specify an action for traffic flows, and defines how to process the traffic [1].

A packet classifier must compare header fields of every incoming packet against a set of rules with the purpose of assign a flow identifier that is applied in security policies [2][3][4].

Providing enhanced data structures, assigning priorities to rules to avoid conflicting and multiple matches, and pre-processing the Rule Base, have been some of the common techniques used by various researchers to improve the algorithms [5]. The basic difficulty of packet classification are large number of rules (size of rule set), growing network traffic (traffic intensity) and large dimensionality of the packet attributes data base (large item sets) [6][7][8][9][10][11][12].

Through this paper, our objective is to modify the existing packet classification system, based on heuristic method, to reach a faster packet classification system and providing a comparative evaluation. It results in a more appropriate and reasonable performance with a logical benchmark.

OpenMP [13] (Open Multi-Processing) is well-thought-out as an execution of multithreading. The threads run parallel with the runtime environment, assigning threads to varied processors. We have used the OpenMP API commands and functions that support C programing language to implement the DimCut and have attempted to forward a new algorithmic technique to solve the packet classification problems.

In this paper we explain briefly our previous proposed DimCut packet classification algorithm and compare it with the HiCut decision tree-based packet classification algorithm, upon which the DimCut is based. The suggested improvements have been validated through simulated trials [2][14].

The present study paper is organized as follows. Section One contains the various algorithms studied to capture the basic ideas, as each one has its own advantages and disadvantages. Section Two explains and studies the HiCut and DimCut algorithms. Section Three deals with the implementation objectives and the directions followed to develop the software. Section Four provides the results of the comparative evaluation on the concerned work, and also analyses the results of our experiments and findings. Section Five provides the final conclusive remarks.

I. RELATED WORK

In the multidimensional packet classification, trade memory is used for better speed and performance. When the number of rules increases, the result is poor, either towards search time or memory usage. Researchers have been attempting to solve these problems in industrial and academic areas [6] and [15]. The packet classification problem still remains a vital challenge in network processing [6][11][15][16][17][18]. A considerable survey on packet classification algorithms is found in the "Survey & Taxonomy of Packet Classification Techniques" paper [5].

The authors, Srinivasan, Varghese, Suri, and Waldvogel, have recommended the Grid-of-Tries and CrossProducting algorithms for packet classification. The data structure of 'Grid of Tries' algorithm is extended to two fields, and uses a decision tree for packet classification on source and destination address prefixes. The authors suggest the CrossProducting solution for multiple fields and bigger classifiers, and also propose a caching technique with the non-deterministic classification time [11].

Baboescu, Singh, and Varghese are researchers who have proposed Extended Grid-of-Tries (EGT) which essentially sustains multiple fields. It is important to note that the EGT alters the switch pointers to become jump pointers that maneuver the search to all feasible matching filters, rather than the filters with the longest matching destination and source address prefixes [19].

Feldman and Muthukrishnan have developed independent field searches on Fat Inverted Segment (FIS) Trees. It provisions a geometric view of the rule set and maps rules in d-dimensional space [18].

Packet classification, when viewed geometrically, uses the construction of data structures and representation of rules. The preprocessing of rule sets uses the strategy of cutting, or projecting, of the multi-dimensional space. The rule set is partitioned and regrouped, so that a packet can quickly identify a reduced rule set that includes the matching rule. Woo's modular packet classification, Multidimensional Cuttings (HyperCut) and Hierarchical Intelligent Cuttings (HiCut), utilizes this approach in algorithms [2][3][19]. The recursive flow classification (RFC) algorithm is competent in reducing the storage. It has been observed that the storage is reduced, while maintaining the high throughput even for larger filter sets. These recommended tradeoffs should be maintained to obtain better performance [11][17].

There is also a packet classification algorithm that uses the tuple space search technique to expresses a Tuple as a pair of prefix lengths, where one hash table stores the filters that belong to one same tuple [20]. The lookup implementation for tuple space techniques differs significantly. Lookups in individual tuple can be conducted through a simple hash table [2].

The bit vector search algorithm [12] is based on a set of rules which matches a packet in a dimension. The characteristics of this technique is that, the packet header can be split into substring and matched with a subset of rules, whose intersection will give a rule to match the whole packet header. Baboescu and Varghese had proposed the Aggregated Bit-Vector (ABV) algorithm that helped to improve the performance function of the Parallel BV technique [12][21].

Every technique possesses some shortcomings, positive and negative aspects, extensions and limitations. To achieve a decent performance, the algorithm has to be designed to combine all methods, useful characteristics and also utilize the time-space tradeoff well.

Many researches have analyzed and explained the problems of packet classification, and several solution algorithms have been proposed, but it still remains a problem, especially for emerging services, which eventually increases the number of classification rules, classification packets and bandwidth [1][22].

II. HICUT AND DIMCUT ALGORITHMS

A. HiCut

Gupta and McKeown have introduced a packet classification algorithm called Hierarchical Intelligent Cuttings (HiCut) [2]. The concept of "cutting" comes from viewing the packet classification problem geometrically. HiCut preprocesses the rule set in order to build a decision tree with leaves containing a small number of rules delimited by a threshold. Packet header fields are used to traverse the decision tree until a leaf is reached. The rules stored in that leaf are then linearly searched for a match. HiCut uses of only four fields (dimension) to construct the decision tree. The algorithm uses various heuristics to select decision principles at each node that minimize the depth of the tree while controlling the amount of memory used. The number of cuts

is determined by the local cutting situation and a configurable space measure factor, spmf. The largest possible number of cuts is taken as long as the following inequality is satisfied.

spmf * number of rules at node r $\geq \sum$ number of rules at each child of node r + number of cuts

Apart from the number of cuts, the dimension to cut along each decision tree node r is also critical to the algorithm performance. The algorithm gives four options. Neither one is consistently better than the others for different rule sets. A larger bucket size can help reduce the size and the depth of the tree, but will take a longer linear search time. Also some optimization can be done like redundancy elimination and child node reusing [6]. The results are not comparable for different rule sets and options. In some cases, some suggested options by the algorithm do not work at all. In practice, the user needs to test all the options and possibilities to find the better proper one [18].

B. DimCut

The DimCut algorithm is equipped modifications and improvements on the HiCut algorithm. DimCut is a Packet Classification Algorithm based on a decision tree, using Recursive Dimensional Cutting. It has two separated levels, pre-processing level (tree construction) and search level. DimCut deals with the geometric view of the packet classification problem, where each rule defines a d-dimensional rectangle in d-dimensional space, and where d is the number of fields in the rule. The algorithm pre-processes the rule set implied to establish a decision tree, and it is well elucidated that the leaves contain a subset of rules with the number of rules bound by a predefined threshold. Packet header fields look up for the proper leaf, and then linearly search for a matched rule belonging to that leaf [14].

The DimCut uses a heuristic to pick an appropriate dimension to cut across and pick the appropriate number of partitions (cut) to be made, with the purpose of distributing the rules inside the partitions properly in a balanced manner, and with minimum possible rules repetition that would be found practically. A larger number of cuts at a node reduce the tree depth, but may increase rule replication and the number of branches, which may not achieve a good rule separation and also increase the memory usage. The process of cutting is performed at each level, and recursively on the child nodes of that level, until the number of rules associated with each node fall below the threshold (maximum number of rules that can be at a leaf node). We have attempted to find heuristics and techniques that can modify the algorithm to reach a high performance with reasonable memory consumption.

The DimCut algorithm provides certain modifications and enhancements on the HiCut algorithm. In DimCut, the GL (H) is the geometric length associated with column H in the whole of the rule set. To choose the proper cut dimension, two fields Ha, Hb are selected which have the least GL ( ) values.

Regression analysis is a statistical process for estimating the relationships among variables. To decide the number of cuts, regression analysis was conducted on 45 different number of rules (100- 100000) with 10 different number of cuts samples. Based on the analysis and results of several tests with reference to efficiency and performance , it is found that the best Number of cuts can be computed with the formula, $NC = 495.22 + (0.034 * N) + (9 * 10^{-7} * N^2) + (6.23*10^{-12} *N^3)$, the Bucket size (The threshold) set as, B = 2 if N<=10000 and B=5 if 10000<N<40000 and B= 8 if 40000<=N<=100000, Here, N = Total Number of rules (as provided some of samples in “fig. 12, 13, 14,” as the specified evidence set out in Appendix A).

In this algorithm, the Array Pointer structure is used which works with a large amount of rules. All rules have been arranged in priority order, in accordance with the network administrator policy. The decision tree will extend across to search the buckets covering the incoming packet and will jump to the first bucket regions of its origin. When the first match bucket is available, a packet will forward to all possible regions of the bucket and then all the header fields of the packet will compare to all governing rules linearly. In this process, the most prioritized rule is selected which matches perfectly.

Rule classification example with 9 rules with 6 fields, is shown in Table I and “Fig. 1, 2, 3”.

“Fig. 1,” shows the geometric view of rules in Table I. In this example, at the first level we select the SrcIP dimension as the cut dimension, the number of cut is set to 4 and the threshold is set to 2, partition one includes R2, R7 and R8, partition two includes R9, partition three includes R5 and R6, and in partition four there are R1, R3 and R4. As the node one and four have more than two rules, so they need to be partitioned again, as shown in “Fig. 2”. The full space range for IP addresses are 0 to 2^{32}, while the ranges of each partition, after cutting, are mentioned in “Fig. 2, 3”.

Rule	Src IP	Dst IP	Src Port	Dst Port	Protocol	Action
R1	192.168.0.1/20	10.0.10.1/8	443	*	TCP	Accept
R2	10.0.0.1/10	192.168.200.100/24	*	1-1024	TCP-UDP	Deny
R3	192.168.200.100/24	172.16.0.1/16	25	1024-65535	UDP	Forward
R4	192.168.100.100/28	172.16.0.1/20	80	*	TCP	Accept
R5	172.16.0.1/16	10.10.100.100/8	1-1024	25	*	Deny
R6	172.16.0.1/20	10.0.10.1/8	8080	*	TCP	Deny
R7	10.0.10.1/8	192.168.0.1/20	21	1-1024	UDP	Accept
R8	10.10.100.100/8	120.0.0.1/24	80	*	TCP	Accept
R9	120.0.0.1/24	172.16.0.1/16	*	1024-655535	*	Accept

Table I: In this example 9 rules are shown in priority, with six fields (source Ip address, destination Ip address, source port, destination port, protocol and action (policy)).

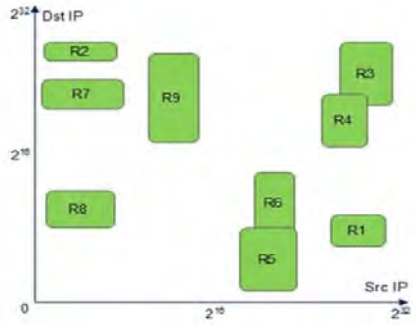

Fig. 1. Each rectangle represents a geometric view of a rule. Whole space should divide into smaller spaces to partitioning the rule set.

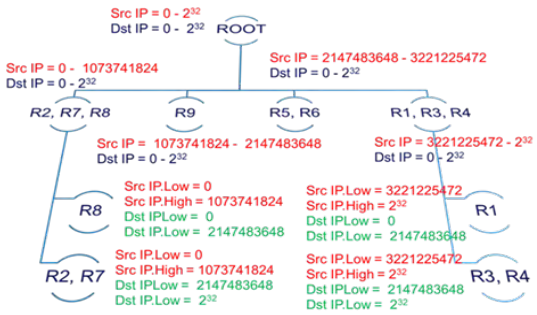

Fig. 2. Shows bucket ranges in cut dimension fields; the root node covers all portions of the d dimensional space (Rule classification).

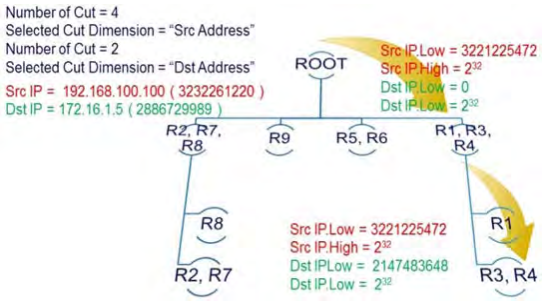

Fig. 3. Shows how the incoming packet finds and jumps to the proper bucket and matched rule.

In the second level of partitioning, we will select the DstIP dimension as the cut dimension, and the number of cut is set to 2. So the nodes are arranged as shown in "Fig. 2". As these nodes do not include more than the threshold amount of rules, so all the nodes are leaf nodes. Therefore, the tree construction (rule classification) gets completed.

With the purpose of classifying any packet, according to the cut dimension corresponding field, the leaf node which is storing the best matching rule for the packet, should be considered. To arrive at the proper leaf (Bucket) by using the following method, it's possible to jump to the proper node rather than traversing the tree, which is the main key for the high performance and efficiency of our algorithm (the mathematically proof is illustrated in Appendix B).

Bucket number = [(Packet (Cut Dimension) – Bucket (Cut Dimension).Low) / ((Bucket (Cut Dimension).High - Bucket (Cut Dimension).Low) / Number of Cuts)] +1

Suppose the incoming packet p1 that has source IP Address 192.168.100.100, destination IP Address 172.16.1.5, source port 80, destination port 1024 and protocol TCP, needs to get classified, as shown in "Fig. 3".

Digital number equal to source address: $192*256^3+168*256^2+100*256+100=3232261220$

Digital number equal to destination address: $172*256^3+16*256^2+1*256+5=2886729989$

We are going to find the node that covers this incoming packet, so according to the Bucket number finding formula:

Selected cut dimension is Source IP Address, Bucket number = $[(3232261220 - 0) / ((2^{32} – 0) / 4)] + 1 = 4$

The next level selected cut dimension is Destination IP Address, Bucket number = $[(2886729989 – 0) / ((2^{32} – 0) / 2)] + 1 = 2$

According to the procedure, the first bucket number will be 4, and as it also has a child, the second bucket number at the next level will be 2. So, there are R3 and R4 in the target bucket and, after linearly search, will find the fully matched rule, that is, R4. Finally, as the R4 action is acceptance in Table I, so the P1 will get accepted.

The index table indexes a reference number to the proper bucket that covers the incoming packet after the optimization, such as eliminating the empty nodes, region compaction, node merging etc.

"Fig. 4," shows all the main procedures for packet classification, and Appendix C shows the Pseudo Codes in "Fig. 15, 16, 17, 18".

It is significant that optimization of the decision tree can be completed by excluding the empty nodes, and combing the nodes that belong to the same set of rules [2][3].

Our efforts have been to search for better methods in order to calculate the number of cuts, selecting the cut dimension and setting the proper threshold along with implementing a suitable technique and data structure to affect the algorithm performance.

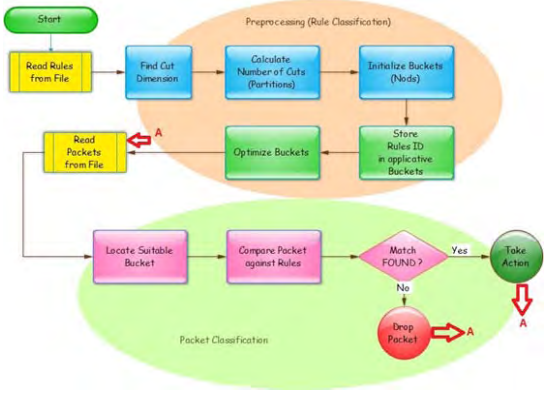

Fig. 4. Shows Classifier flowchart and main procedures.

III. EXPERIMENTAL METHODOLOGY

Contribution of our work lies in the detailed and consistent evaluation of such selected algorithms that have been implemented with common principles and evaluated in a common test bed.

To compare Hicut and DimCut (simple mode and using six threads mode) with each other, we have run the full test, provided the data analysis and drawn the suitable graphs. Search performance is evaluated by running it through large number of packets and rules; and to reach the best evaluation, the worst case scenario is used for providing the same condition for all tests.

The testing experiments have been conducted on standard PCs with 8 cores Intel Xeon 3.00 GHz, RAM 8.00 GB, using the Oracle VM Virtual Box to provide an isolated environment, using GCC 4.7.1 compiler, with OpenMP enabled and OpenMP disabled.

For these tests, the 1000, 5000, 10000 … 100000, numbers of random rules and the 20000 numbers of random packets have been generated, the packet size is 20 Byte and the rule size is 52 Byte.

All tests have been conducted multiple times and all tests run in two execution models, Normal Run (Single Thread) and OpenMP (Multi Threads - Parallel). The OpenMP is an API that supports multi-platform shared memory multiprocessing programming in C and it's an implementation of multithreading, a method of parallelizing by using Omp.h header file, gomp and pthread library.

In the"Fig. 5," the plan of the simulated experimented is shown.

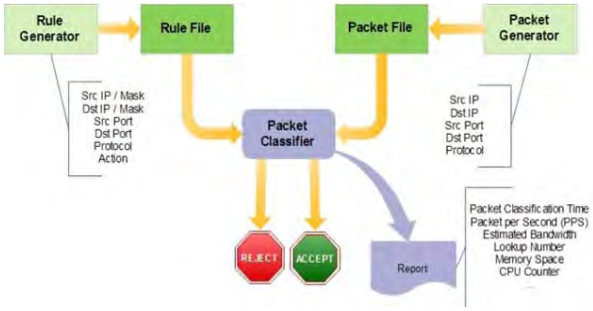

Fig. 5. Simulated experimented methodology model setup.

The rule's headers are Source IP (32 bit) and Destination IP (32 bit): (Exact/prefix), Source Port (16 bit) and Destination Port (16 bit): (Exact value, any, ranges), Protocol (8 bit): (TCP, UDP, ICMP, ANY, IGMP, GRE, IGP, EGP) and the Actions (8 bit): (Accept; Deny, Log, Forward, Nothing).

The evaluation metrics and parameters constitute the Cut Dimension, Number of Cut, Bucket Size, Rule Classification Time (the amount of time needed to classify rules), Packet Classification Time (the amount of time needed to classify packets), PPS (Packet Processed per Second), Bandwidth (Mbps) = (PPS * 8 * (size of (Packet)) / (1024 * 1024)), Rule Memory Access, Bucket Memory Access (which shows the amount of read or write a number of bytes to or from memory during packet classification process among the Buckets), RTSC (Read Time Stamp Counter or number of CPU clock cycles ticks from the machine bootstrap), Number of Bytes Accessed per Packet =((Rule Memory Access + Bucket Memory Access) / Packet Count), and Memory consumption (the amount of maximum memory usage at the run time for both rule and packet classification).

The RDTSC ("read time stamp counter") instruction is available on processors and it is a tool for accurate timing. It stores the number of elapsed clock-cycles since the time the processors are powered on. So, by comparing the results of RDTSC before and after some action, could give the real run-time timing information accurately to the clock cycles.

It is notable here that our reference implementations are only for the purpose of simulation and evaluation; thus, the source code is not optimized as software. We have selected the configurations that lead to the best overall performance.

IV. EXPERIMENTAL RESULTS

Graphs

The following graphs Show the comparison between the HiCut, the DimCut and the DimCut running with six threads (T6). The "Fig. 6,"shows that, while the rules are increasing the Packet classification time also increases. The DimCut T6 acts better with respect to time consumption and it is faster than the others during the course of packet classification process.

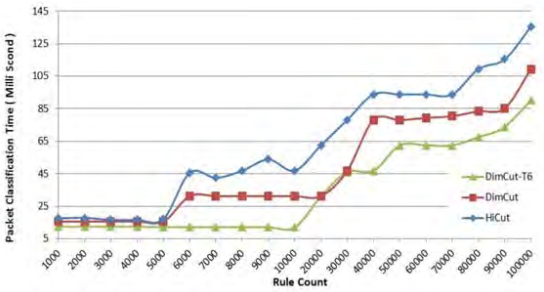

Fig. 6. A comparison between the HiCut, DimCut and DimCut T6, to measure the packet classification time (milli second).

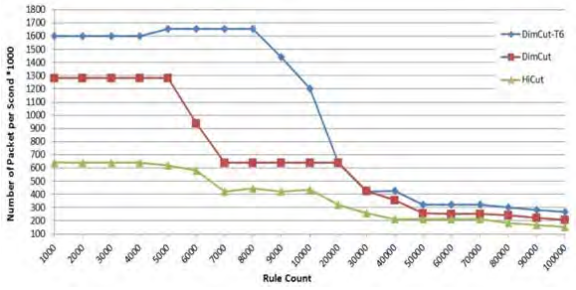

Fig. 7. A comparison between the HiCut, DimCut and DimCut T6, to measure the number of packet per second processing.

The “Fig. 7,”shows the Packet per Second processing (PPS), where, as the rules number increases, the numbers of packet processing ability is decreasing during the packet classification action. However, the DimCut algorithms seem to be more efficient than others. Memory usage is far higher in case of the HiCut algorithm when compared to DimCut algorithms that their memory consumption grows linearly with the number of rules, as can be seen from “Fig. 8”. The DimCut maximum memory consumption according to this test setup, for at least 100000 rules would be near to 15 MB and that is very reasonable amount.

The “Fig. 9,”shows the estimated bandwidth measurement, which explains DimCut efficiency and performance. It shows the Mega bit packet data processing per second. The “Fig. 10,”shows the number of Bytes Access per Packet, which counts all the numbers of memory in Byte that are accessed during packet classification per packet. The “Fig. 11,”shows the Time Stamp Counter per Packet, which counts the number of cycles for each packet during the packet classification. The number of cuts and the dimension selection to cut at each internal decision tree node are the critical sensitiveness for the algorithm performance. A larger bucket size or lesser number of cuts can help to reduce the size and depth of a decision tree, but it can induce a longer linear search time. Experimentally could determine the appropriate bucket size for the best trade off of storage and throughput.

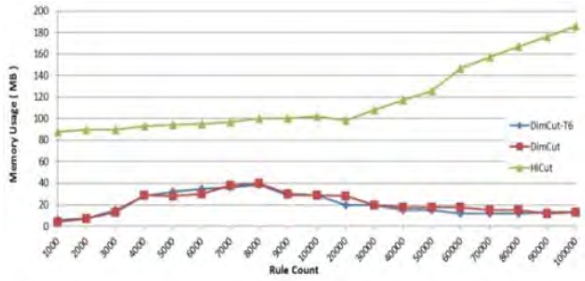

Fig. 8. A comparison between the HiCut, DimCut and DimCut T6, to measure the maximum of memory usage (M).

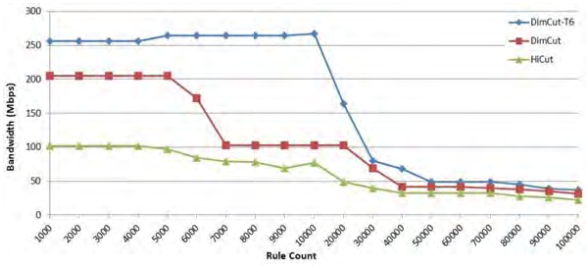

Fig. 9. A comparison between the HiCut, DimCut and DimCut T6, to measure the Bandwidth (Mbps).

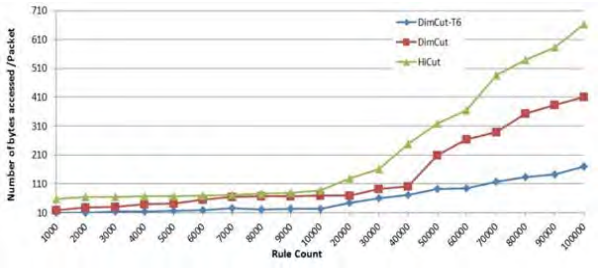

Fig. 10. A comparison between the HiCut, DimCut and DimCut T6, to measure the Number of Bytes Access / Packet.

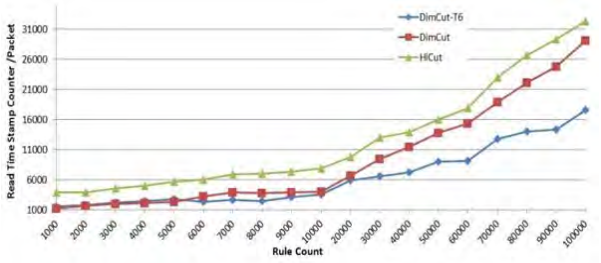

Fig. 11. A comparison between the HiCut, DimCut and DimCut T6, to measure Time Stamp Counter per Packet, which counts the number of cycles for each packet.

The performance studies show that DimCut can provide an improvement up to 74% of the given rules in Packet Classification Time calculation rather than the HiCut, as can be seen from Table II.

The outcomes of this test confirm that the HiCut is slower than DimCut. The HiCut algorithm shows the tendency is approximately linear on the number of rules up to 10000 rules. In case of the memory usage, the HiCut's memory consumption is up to 15 times more than the DimCut algorithm.

Rule Counts	DimCut T6	DimCut
1000	29%	11%
5000	29%	8%
10000	74%	33%
50000	33%	16%
100000	33%	19%

Table II. The percentage of improvement in Packet Classification Time is shown rather than HiCut, for the given Rules in worst case scenario.

According to the data analysis and graphs, it is proved that the proposed algorithms based on decision tree, make packet classification faster, as compared to HiCut algorithm.

V. CONCLUSION

This paper focuses on the evaluation issues for high performance packet classification algorithms, which is an important factor in Firewalls, routers, network security and quality of service (QoS) assurance.

Main contribution of our work lies in the detailed and consistent evaluation of HiCut and DimCut classification algorithms that have been implemented with common principles and evaluated in a common test bed, by measuring the Packet Classification Time, Number of Packet per Second Classification, Rule Memory access, Preprocessing Time/Tree Construction Time, Number of buckets (leaves), Depth of the tree structure and Threshold.

We utilized multi-threading by using OpenMP to implement the DimCut and achieved better performance and results.

The final results have been illustrated in graphics for better representation. We are of the view that DimCut can be a viable Packet Classification algorithm that gives a deterministic performance, besides providing flexibility for system designers to tradeoff the components, and thus benefit the research and design community as a whole.

Further studies are, therefore, required to explore more systematic ways for perfecting the configurable parameters and improving the adaptive decision-tree construction procedures.

ACKNOWLEDGMENT

The authors express their sincere appreciation to the anonymous reviewer(s) for their insightful comments and help rendered to improve the paper's quality of presentation.

REFERENCES

[1] M. Sundström, "Time and Space Efficient Algorithms for Packet Classification and Forwarding", Doctoral Thesis, Luleå University of Technology Department of Computer Science and Electrical Engineering Centre for Distance Spanning Technology, 2007.
[2] S. Singh, F. Baboescu, G.Varghese and J.Wang, "Packet Classification using Multidimensional Cutting", in Proceedings of the ACM SIGCOMM '03 Conference on Applications, Tech., Archi., and Protocols for Computer Communication (SIGCOMM '03), pp.213 – 224, 2003.
[3] P. Gupta, N. McKeown, "Packet Classification Using Hierarchical Intelligent Cuttings", in Proceedings of IEEE Symp. High Performance Interconnects (HotI), 7, 1999.
[4] B. Vamanan, G.Voskuilen, T.N. Vijaykumar, " EffiCuts: optimizing packet classification for memory and throughput", in Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, New Delhi, India, 2010.
[5] D. Taylor, "Survey & Taxonomy of Packet Classification Techniques", in Proceedings of ACM Computing Surveys (CSUR), vol 37, Issue 3, pp. 238 - 275, September 2005.
[6] H. Song, J. Turner, "Toward Advocacy-Free Evaluation of Packet Classification Algorithms", in IEEE Transactions on Computers, vol. 60, MAY 2011.
[7] M. Bauer, "Paranoid penguin: Using Iptables for local security", in Linux Journal, Available at http://www.linuxjournal.com/article/609, August 2002.
[8] D. Napier, "IPTables/NetFilter – Linux's next generation stateful packet filter", in Sys Admin - Security: The Journal for UNIX Systems Administrators, 10(12):8, 10, 12, 14, 16, December 2001.
[9] T.Woo, "A Modular Approach to Packet Classification: Algorithms and Results", in Proceedings of INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol.3, pp. 26-30 Mar 2000.
[10] D. Decasper, Z. Dittia, G. Pantlkar and B. Plattner Scottberg, "Router plugins: A software architecture for next generation routers", in Proceedings of ACM Sigcomm, pp. 191-202, Vancouver, Canada, 1998.
[11] V. Srinivasan, G.Varghese, S. Suri and M. Waldvogel, "Fast and scalable layer four switching", in Proceedings of ACM Sigcomm '98, pp. 191-202, Vancouver, Canada, 1998.
[12] D. Stihdis, T.V. Lakslunan, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching", in Proceedings of ACM Sigcomm, pp. 203-214, Vancouver, Canada, August 31 – September 1998.
[13] OpenMP Architecture Review Board, 2013, Available at http://en.wikipedia.org/wiki/OpenMP and http://openmp.org/wp/
[14] H. Amirjahanshahi, M. Poustchi, H. Acharya, "Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut)", in proceeding of the Research journal of Recent Sciences, 2(8), pp.31-39, August 2013.
[15] M. Waldvogel, G. Varghese, J. Turner and B. Plattner, "Scalable High Speed IP Routing Lookups", in Proceedings of the ACM SIGCOMM, 25-38, 1997.
[16] V. Srinvasan, and G. Varghese," Fast Address Lookups Using Controlled Prefix Expansion", in Proceedings of the ACM Transactions on Computer Systems, Sigmetrics '98/Performance'98 Joint International Conference on Measurement and Modelling of Computer Systems, 1999.
[17] P. Gupta and N. McKeown, "Packet Classification on Multiple Fields", in proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication in ACM SIGCOMM '99, 147-160, 1999.
[18] A. Feldmann and S. Muthukrishnan, "Trade-offs for Packet Classification", in Proceedings of the IEEE INFOCOM, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 3, 1193–1202, 2000.
[19] Y. Qi and J. Li, "An efficient hybrid algorithm for multidimensional packet classification", in Proceedings of the International Conference Communication, Network, and Information Security, MIT, Cambridge, MA, USA, October 9 – 11, 2006.
[20] H. Song, J. Turner and S. Dharmapurikar, "Packet Classification Using Coarse-Grained Tuple Spaces", in Proceedings of the ACM/IEEE Symp, Architecture for Networking and Comm., Systems (ANCS '06), 41- 50, 2006.
[21] F. Baboescu and G. Varghese, "Scalable Packet Classification," ACM SIGCOMM, 2001.
[22] M. Abdelghani, S. Sezer, E. Garcia and M. Jun, "Packet Classification Using Adaptive Rules Cutting (ARC)", in Proceedings of the IEEE Telecommunications, advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop, 2005.

Appendix A

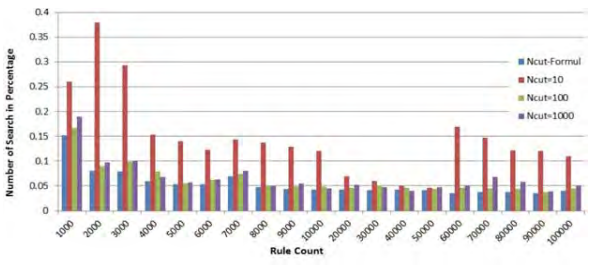

Fig. 12. Test the varying number of cuts behavior to find the better amount. In contrast, points those match with the proposed NC formula, almost shows lower number of search processing across increases in rules.

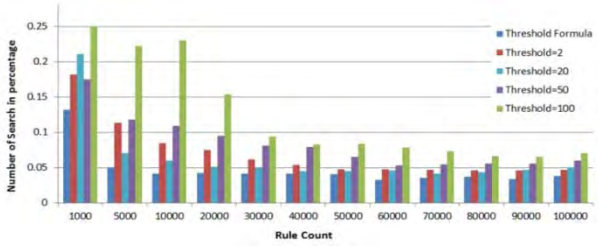

Fig. 13. Test the varying Threshold (Bucket Size) behavior to find the better amount. In contrast, points those match with the proposed Threshold set, almost shows lower number of search processing across increases in rules.

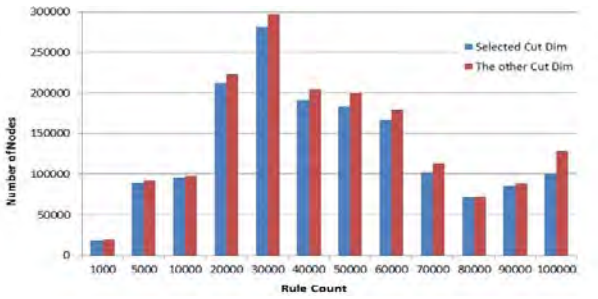

Fig. 14. It Shows the Cut Dimension selection differences. In contrast, points those match with the proposed Cut Dimension selection, almost shows lower number of nodes to construct the tree across increases in rules.

Appendix B

PROOF:

Since there is exactly one integer in a half-open interval of length one, for any real x there are unique integers m and n satisfying

$x-1 < m <= x <= n < x+1$ Then

$[x] = m$ and $[x] = n$ may also be taken as the definition of floor and ceiling.

So

1. Let x is a real number, greatest integer number is a step function written as f(n)=[x], where f(n) is the greatest integer less than or equal to x. [1.99]=1
2. Properties of G.I.F. Let x is a real number, then
   a) $x-1<[x]<=x$
   b) $[x]<=x<[x]+1$
3. Definition
   Let x and y are two real numbers x, y $\in$|R and n is a natural number n $\in$|N, Then interval [x,y) can be divided to N subinterval as $[x_{i-1}, x_i)$, i= 1, 2, … Where

$x_0=x$, $x_n=y$, … , $x_i=x_0 + ih$ where $h=((y-x) / N)$, then

$\{ x= x_0<x_1<x_2<\ldots< x_{n-1}<x_n = y \}$ is a partition of [x, y).

$[x, y) = [x_0, x_1) \cup [x_1, x_2) \cup \ldots [x_{n-1}, x_n)$

$= U^n_{i=1}[x_{i-1}, x_i)$

4. Definition
   Let N_i is an arbitrary number in i th subinterval $[x_{i-1} , x_i)$ then

$x_{i-1} <= N_i < x_i$

Theorem

Let x, y and N_i are arbitrary real number those designed in definition 4 , then

$i = [((N_i – x) / ((y – x) / N)) +1]$

Proof :

Ni is in ith subinterval then,

$N_i \in [x_{i-1}, x_i)$ or $x_{i-1} <= N_i < x_i$ then

$X_0 + (i-1)h <= N_i < x_0 + ih$

$X + (i-1)h <= N_i < x + ih$

Since $x_0 = x$, then

$(i-1)h <= N_i - x < ih$

$(i-1) <= (N_i - x) / h < i$

$i <= ((N_i - x) / h) + 1 < i+ 1$

Using properties 2.a , one can write,

$[((N_i - x) / h) + 1] = i$

But $h = ((y - x) / N)$, then

$i = [((N_i - x) / ((y - x) / N)) +1]$

i = Bucket No. that incoming packet is belongs to

N_i = Packet (Cut Dimension) field

N= number of Cuts or partitions

X= Bucket (Cut Dimension).Low

Y= Bucket (Cut Dimension).High

Bucket number = [(Packet (CutDimension) – Bucket (CutDimension).Low) / ((Bucket (CutDimension).High – Bucket (CutDimension).Low) / Number of Cuts)] +1

Appendix C

```
SET G_SRC and G_DST to 0
FOR each rule in RULE ARRAY
    COMPUTE GEOMETRIC_LENGHT_SRC as subtract of broadcast and network
    source IP address
    COMPUTE GEOMETRIC_LENGHT_DST as subtract of broadcast and network
    destination IP address
    COMPUTE G_SRC as sum GEOMETRIC_LENGHT_SRC and G_SRC
    COMPUTE G_DST as sum GEOMETRIC_LENGHT_DST and G_DST
END FOR
IF  Minimum of G_SRC and G_DST = G_SRC THEN
    SELECT source address as cut dimension
ELSE
    SELECT destination address as cut dimension
END IF
```

Fig. 15. Find Cut Dimension – Pseudo Code.

```
CREATE array BUCKET[NC] //NC=number of cut
COMPUTE r as 2^32 divided by NC
SET low to 0
FOR each bucket in BUCKET ARRAY
    SET high to SUM low and r
    SET bucket.Field[ cut dimension ].low to low
    SET bucket.Field[ cut dimension ].high to high
    COPY ALL rule numbers to bucket
    SET low to high
END FOR
CALL SplitBucket with BUCKET ARRAY
```

Fig. 16. Initialize Buckets (nodes) – Pseudo Code.

```
FOR each bucket in BUCKET ARRAY
    For each rule in bucket
        IF rule does not belong to bucket
        REMOVE rule from bucket
    END FOR
    WHILE number of rules in bucket is greater than THRESHOLD
        REVERSE cut dimension
        SPLIT bucket into number of cut
        For each rule in split bucket
            IF rule does not belong to bucket
            REMOVE rule from bucket
        END FOR
    END WHILE
END FOR
CALL OptimizeBucket with BUCKET ARRAY
```

Fig. 17. Split Buckets (nodes) – Pseudo Code.

```
WHILE rule exist in BUCKET
    IF packet fields match all rule fields THEN
        //Packet FOUND
        DO rule.field[ action ]  //ACCEPT/REJECT
        RETURN
    END IF
END WHILE
// Packet NOT FOUND
DROP packet
RETURN
```

Fig. 18. Search Packet – Pseudo Code.