Comparative Evaluation of Packet Classification Algorithms, with Implementation

Hediyeh AmirJahanshahi Sistani[1], Haridas Acharya[2]

[1]Department of Computer Studies and Research, Symbiosis International University, Pune, India
hediehamirjahanshahi@yahoo.com

[2]Allana Institute of Management Science, Pune University, Pune, India

harydas.undri@gmail.com

Abstract—in a realm of ever-increasing Internet connectivity, together with swelling computer security threats, security-cognizant network applications technology is gaining widespread popularity. Packet classifiers are extensively employed for numerous network applications in different types of network devices such as Firewalls and Router, among others. Appreciating the tangible performance of recommended packet classifiers is a prerequisite for both algorithm creators and consumers. However, this is occasionally challenging to accomplish. Each innovative algorithm published is assessed from diverse perceptions and is founded on different suppositions. Devoid of a mutual foundation, it is virtually impossible to compare different algorithms directly. In the interim, it too aids the system implementers to effortlessly pick the most suitable algorithm for their actual applications.

Electing an ineffectual algorithm for an application can invite major expenditures. This is particularly true for packet classification in network routers, as packet classification is fundamentally a tough problem and all current algorithms are constructed on specific heuristics and filter set characteristics. The performance of the packet classification subsystem is vital for the aggregate success of the network routers.

In this study, we have piloted an advanced exploration of the existing algorithms to provide a comparative evaluation of a number of known classification algorithms that have been considered for both software and hardware implementation. We have explained our earlier suggested DimCut packet classification algorithm, and related it with the BV, HiCuts and HyperCuts decision tree-based packet classification algorithms with the comparative evaluation analysis. This comparison has been carried out on implementations based on the same principles and design choices from different sources. Performance measurements have been obtained by feeding the implemented classifiers with a large number of random Rules and Packets in the same test scenario.

Keywords—
Firewalls; Packet Classification; Bit Vector; HyperCuts; HiCuts; DimCut

INTRODUCTION

Packet classifiers are extensively used for a variety of network applications, of which several are correlated to quality of service (QoS) provisions, and therefore in numerous varieties of network devices. Large scale packet classification has become a key element of network security systems and Firewalls needing to classify packets, where the speed of decision making, to accept or reject, is of utmost significance.

The foremost task of a firewall is to scrutinize and select the network traffic in accordance with the designated security policy. Typically, the security policy and rules are encoded manually by a system administrator to stipulate an action for traffic flows, and, therefore, specify how to process the traffic. The security policy system spells out the progression of the network traffic. Packet classifiers are extensively employed in IP networking where procedures customarily comprise one or more packet header fields. A packet classifier must compare header fields of each inward packet against a set of rules [1][2][3][4]. The packet header fields generally consist of the source IP address, the destination IP address, the transport protocol, the source port, and the destination port, which can be an exact prefix and range. Each filter R[i] has an allied action that governs how a packet P is handled if P matches R[i]. Filters can overlap; hence, a packet can match multiple filters, but the one with the highest priority among all the equivalent filters is selected as the best matching filter. Customarily, the filter's position in an ordered list of filters defines its priority [1][5].

Another feature that can assist a faster packet classification is using multi-processing and multi-threading. OpenMP (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 called it the DimCut T [6].

In this study, we have elucidated our erstwhile recommended DimCut packet classification algorithm, and compared DimCut and DimCut T with the HiCuts, HyperCuts and BV decision tree-based packet classification algorithms. The proposed improvements have been corroborated by simulated trials.

The rest of the paper is organized as follows: Section One reviews some related works on the various algorithms studied to capture their advantages and disadvantages. Section Two explains the BV, HiCuts, HyperCuts, DimCut and DimCut T algorithms. Section Three deals with the implementation objectives. Section Four tabulates and examines the results of the comparative assessment of our experiments and findings. Section Five is a summary of our contributions and conclusion.

I. Literature Review

A packet classifier must correlate header fields of all incoming packets against a set of rules containing the security policies. Packet classification aims at pursuing the best matching filter for a given packet header. In the multidimensional packet classification, trade memory is used for better speed and performance. When the number of rules increases, the result is inadequate, either towards search time or memory usage [7][8][9][10][11][12][13].

The bit vector search algorithm is derived on the basis of filter set intersecting, as it is easier to match a part of filter at a time than to match the entire filter all together. The packet header can be split into substrings and matched with a subset of rules, whose intersection will give a rule to match the whole packet header [14].

Baboescu, Singh, and Varghese are researchers who have proposed Extended Grid-of-Tries (EGT) which fundamentally withstands multiple fields. It is notable that the EGT modifies the switch pointers to jump pointers that maneuver the search to all feasible matching filters, rather than only to the filters with the longest matching destination and source address prefixes [15]. Woo's modular packet classification, Multidimensional Cuttings (HyperCuts) and Hierarchical Intelligent Cuttings (HiCuts), employs filter set splitting method in algorithms, where the preprocessing of rule sets utilizes the strategy of cutting of the multi-dimensional space recursively to construct the decision tree [1][12][15].

Tuple Space Search (TSS) is based on filter set grouping, where filters in a set are rearranged into separate subsets with definite common features. Lookups can be conducted in each of these slighter subsets in parallel. The proper match is extracted from the results of all the lookups. Lookups in each tuple can be organized through a basic hash table [16][17].

Many researchers have examined and illustrated the problems of packet classification, and several solution algorithms have been suggested, but it still remains problematic, leaving innumerable opportunities to improve algorithm performance in the existing algorithms [4][18][19].

II. Bit Vector, HiCuts, HyperCuts and DimCut Algorithms

A. Bit Vector

The linear understanding of packet classification divulges some basic concepts in what manner the data configurations can be created and how to characterize packet filters. In geometric view, several algorithms take on either 'cutting' or 'projection' techniques in multidimensional space to preprocess filter sets. The 'cutting technique' portions the space into smaller sections at designated vantage points. This procedure helps to contract the latitude of the search. The 'projection technique' plans the end-points of ranges to each dimensional axis. Two contiguous points outline an elementary intermission that is completely encompassed by a distinctive subset of filters. 'Projection technique' has advanced granularity than the 'cutting technique' and can, therefore, distinguish filters in a superior manner. However, locating an elementary interval by this technique is more challenging than tracing a sub-region by the 'cutting' technique. The decision tree-based algorithms typically apply the 'cutting' technique, while the decomposition-based algorithms customarily utilize the 'projection' technique.

The Bit Vector (BV) algorithm is a decomposition-based algorithm that characterizes the subset of filters for each partial match by means of bit vectors. The filter set intersecting notion is that it is easier to match a partial filter, rather than the entire filter, at one time. Therefore, when the packet header is divided into a set of substrings, then each substring can match a subset of filters. The intersection of these subsets is precisely the filters matching the total packet header [14].

Consider the example in 2D filter set; shown in "Fig. 1", each filter appears to be a rectangle on this 2D plane. The preprocessing step of the algorithm projects the edges of the rectangles to the corresponding axis, means project the end points of each rectangle to the axis, and any two adjacent projection points on an axis defines an elementary interval which is fully covered by a set of filters. In the example shown, the three rectangles create six intervals in each axis. Then associate a bitmap with each dimension. A bit in the bitmap is set, if the corresponding rectangle overlaps with the interval that the bitmap corresponds to. Since there are 3 filters in total, each bit vector is 3-bit wide. The bit 0 is for the filter R1, the bit 1 is for the filter R2, and so on, as is shown in “Fig. 1”. To implement the BV, we used the binary search technique to build the single field lookup data structure for retrieving the bit vectors.

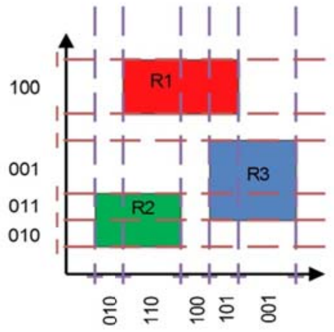

Figure 1. Shows the Projection partitiong techniqe

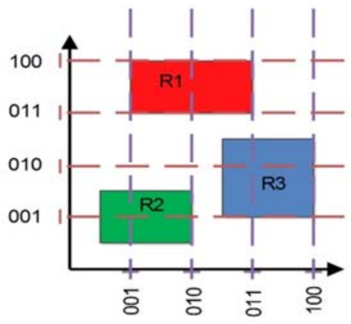

Figure 2. Shows the Cutting partitiong techniqe

B. HiCuts

The packet classification algorithm, Hierarchical Intelligent Cuttings proposed by Gupta and McKeown. The concept of “cutting” emanates from observing the packet classification problem geometrically as shown in “Fig. 2”. HiCuts preprocesses the rule set for constructing a decision tree, with its leaves encompassing a specific number of rules. The number of cuts is determined by the local cutting circumstances and a global configurable space measure factor, spmf. The largest possible number of cuts is chosen, as long as the following inequality is satisfied [1].

spmf * number of rules at node r ≥ ∑ number of rules at each child of node r + number of cuts

The threshold is the maximum number of rules allowable in a leaf node. A larger threshold can assist in decreasing the size and depth of the decision tree, but will take a longer linear search time. In practice, the user needs to explore all options and possibilities for identifying the appropriate one [13][8].

C. HyperCuts

HyperCuts is a decision tree-based packet classification algorithm similar to HiCuts with some differences. The algorithm recursively cuts the space into smaller areas, on multiple dimensions per step. Each smaller area ends up with fewer overlapped hyper cubes which to a point allow a smaller linear search to find the best matched rule. HyperCuts allows cutting on multiple dimensions rather than only one dimensionvin HiCuts, and also it presents some additional optimization improvements such as: node merging, rule overlap, region compaction, pushing common rule subsets upwards [1].

To decide about the number of cuts, assume each chosen dimension would receive nc(i) cuts, then the possible number of child nodes, NC, is Π nc(i). It chooses the largest possible NC that is bound by spfac*sqrt (N), where spfac is a configurable parameter, N is the number of rules of the specific node. The bucket size decides that when should terminate the decision tree construction. A larger bucket size can help decrease the size and the depth of the decision tree, but it cause to a longer linear search time.

D. DimCut

The DimCut algorithm is fortified with certain alterations and enhancements on the HiCuts algorithm. It has two disconnected levels, pre-processing level (tree construction) and search level. The algorithm pre-processes the rule set inferred to create a decision tree, and it is well expounded that the leaves contain a subset of rules with the number of rules bound by a predefined threshold. Packet header fields hunt for the proper leaf, and then linearly search for a corresponding rule fitting to that leaf [5] “in press” [20].

The DimCut utilizes a heuristic for selecting the correct dimension to scan and pick the appropriate number of partitions (cut) to be made, with the objective of distributing the rules inside the partitions in a balanced manner, and with minimum repetition of feasible rules. The process of cutting is implemented at each level, and recursively on the child nodes of that level, until the number of rules linked with each node become lower than the threshold (maximum number of rules that can be at a leaf node). We have endeavored to discover heuristics and techniques that can transform the algorithm for a higher performance with equitable memory consumption.

In DimCut, the GL (H) is the geometric length associated with column H in the rule set. To choose the best cut dimension, two fields Ha, Hb are selected which have the least GL ( ) values. Statistical regression analysis is used to estimate the best number of cuts. Based on 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 + (.034 * N) + (9 * 10-7 * N2) + (6.23*10-12 *N3), 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.

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. To arrive at the proper node 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 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.

III. EXPERIMENTAL METHODOLOGY

Our basic involvement in this work is an unprejudiced comparison with shared standards and valuation circumstances, by giving a homogeneous review of these three classification algorithms which have been executed with common principles and evaluated in a common trial environment.

All the trials have been steered on standard PCs with 8 cores Intel Xeon 3.00 GHz, RAM 8.00 GB, using the Oracle VM Virtual Box to provide an insulated background, using GCC 4.7.1 compiler. Search performance is evaluated by directing it through a large number of packets and rules; and to reach the best valuation, the worst case scenario is used while providing identical settings for all tests.

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

In the "Fig. 3", the blueprint of the simulated experiment is shown.

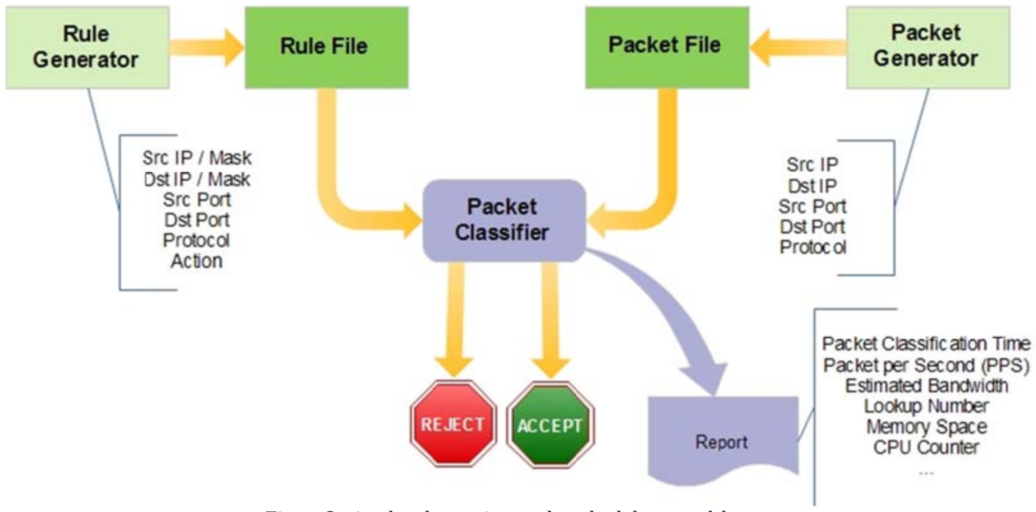

Figure 3. simulated experimented methodology model setup.

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 institute the Cut Dimension, Number of Cut, Rule Classification Time, Packet Classification Time, Rule Memory Access, Number of Search, RTSC (Read Time Stamp Counter or number of CPU clock cycles ticks from the machine bootstrap).

It is significant to note that our cited applications are only for the purpose of replication and assessment, and therefore the source code is not augmented as software. We have prudently picked the formations that show the way to the best inclusive accomplishment.

IV. EXPERIMENTAL RESULTS

The worst case scenario is provided for tests and tests have been conducted multiple times to calculate the average. The DimCut tests run in two execution models, Normal Run (Single Thread) and OpenMP (Multi Threads - Parallel) that called the DimCut T. 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.

The following graphs display evaluation result of the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector packet classification algorithms.

“Fig. 4”, presents that, while the rules are increasing the rule classification time also increases. The both DimCut act better with respect to time consumption and DimCut is faster than the others during the progress of rule classification process. The HyperCuts rule classification time consumption is more reasonable than the HiCuts and BV.

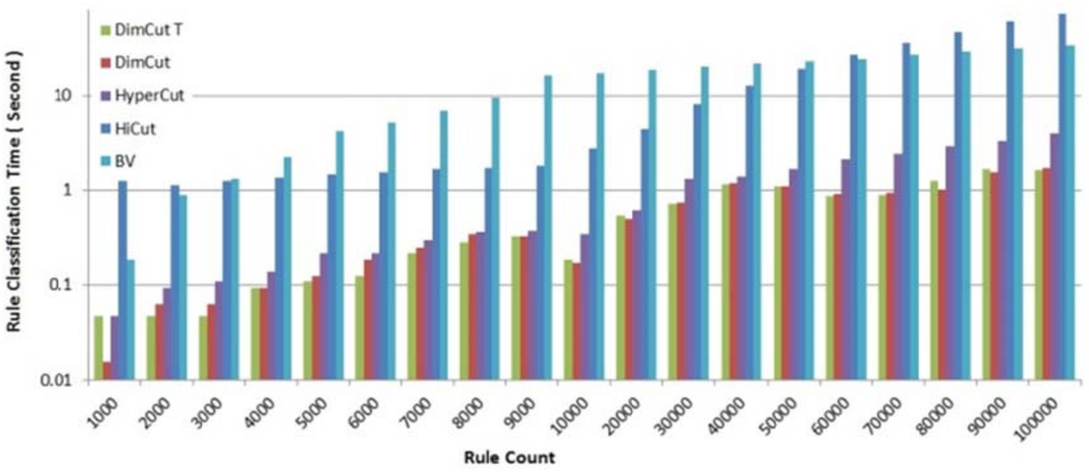

Figure 4. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure the rule classification time (second).

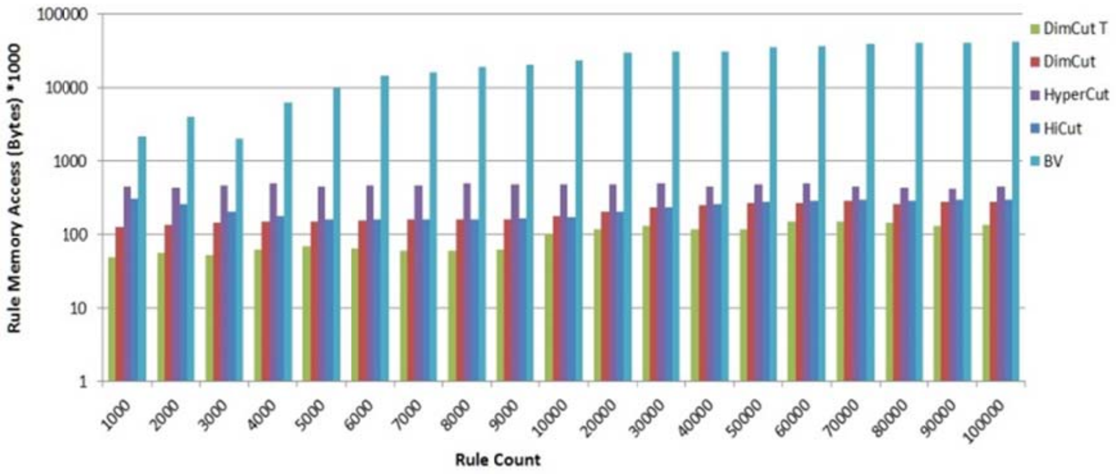

Figure 5. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure the rule memory access (Byte).

“Fig. 5”, shows the rule memory access, where, as the rules number increases, the number of memory access bytes is increases. However, the DimCut algorithm appears to be more competent than others. The HiCuts and HyperCuts are reasonable in case of rule memory access measurement, but the BV has the worst action. “Fig. 6”, shows when the rules increase the packet classification time also increases, but the DimCut T is acting better with respect to time consumption. DimCut T and DimCut are faster than other ones; the BV is slower than the others during packet classification action. “Fig. 7”, shows the comparison in case of the number of CPU time stamp per packet during classification process, which counts the number of cycles for each packet during the packet classification.

“Fig. 8”, shows the comparison in case of total number of search during packet classification process. In case of total number of searches the HyperCuts is acting well, the HiCuts is acceptable but the BV is the worst and the both DimCuts have lesser total number of searches, which explains DimCut efficiency and performance that need to search lesser number of rules during packet classification process.

Memory usage is far higher in case of the HiCuts and Bit Vector algorithms when compared to HyperCuts and DimCut algorithms, as can be seen from “Fig. 9”. The DimCut maximum memory consumption according to this test format, for at least 100000 rules, would be near to 15 MB which is a very equitable amount.

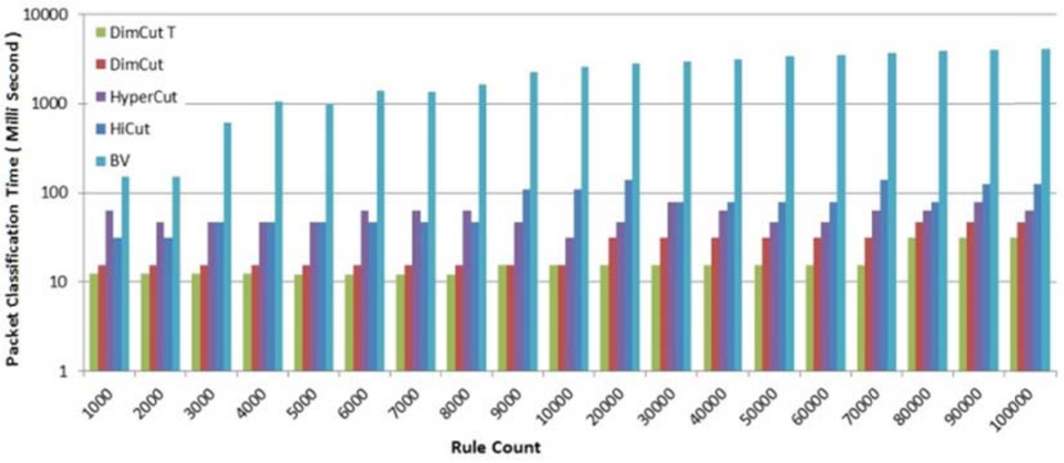

Figure 6. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure the packet classification time (Milli second).

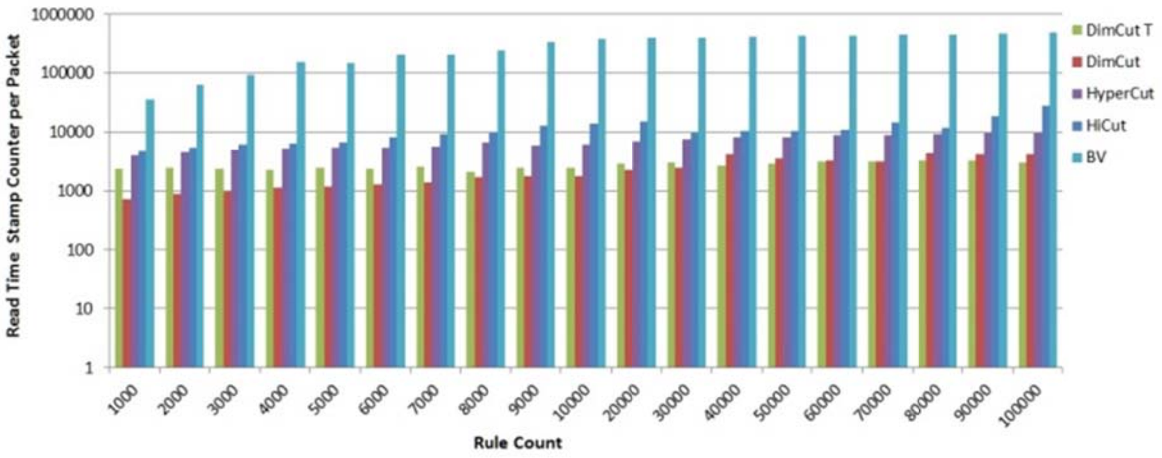

Figure 7. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure the RTSC per packet.

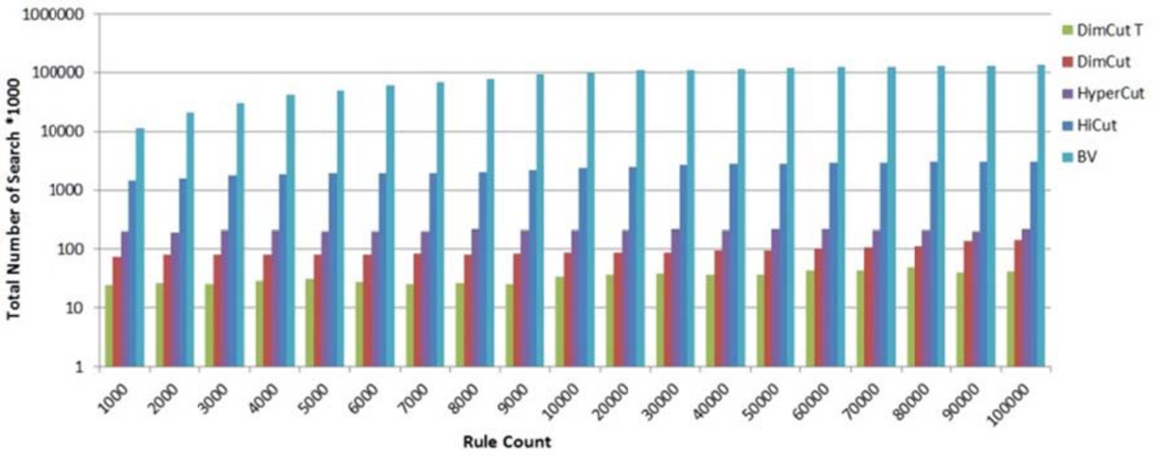

Figure 8. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure the total number of search.

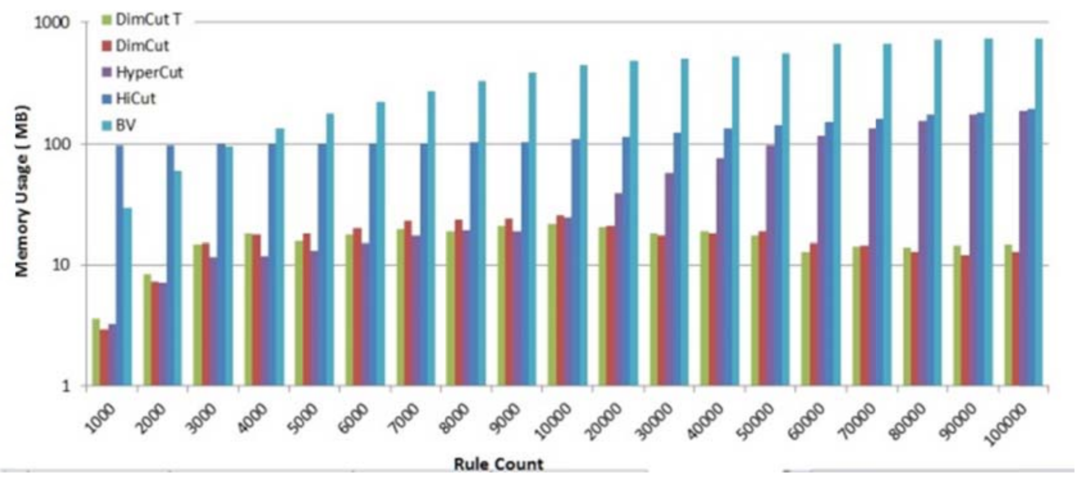

Figure 9. A comparison between the HiCuts, HyperCuts, DimCut, DimCut T and the Bit Vector, to measure of memory usage (M)

The number of cuts, and the dimension selection to cut at each internal decision tree node, is the key criterion for the HiCuts, HyperCuts and DimCut algorithms performance. A larger bucket size, or lesser number of cuts, can enable reduction of the size and depth of a decision tree, but it can provoke a longer linear search time. Experimenting could determine the appropriate bucket size for the best trade off of storage and throughput. Generally, a larger bucket size means a worse search processing but this does not always sustain. According to the above tests, it's clear that the BV algorithm performance is relatively insensitive to the number of rules. Since each Bit Vector is n number of bits equal to number of rules (each bit represent a rule), and each field should make a binary search tree, a very long bit vector needs more time and memory consumption.

The performance studies show that DimCut can provide an improvement of up to 99.1% of the given rules in Read Time Stamp Counter per Rule calculation over the HiCuts, 99.3% over the Bit Vector and 80.2% over the HyperCuts as can be seen from Table 1.

TABLE 1. The percentage of improvement of DimCut rather than HiCuts, HyperCuts and BV for the given Rules in worst case scenario.

Rule Counts	Read Time Stamp Counter per Rule		
	HiCut	HyperCut	Bit Vector
1000	99.13821855	80.25957701	94.29717247
5000	91.10609912	42.00853274	96.91425508
10000	91.38404495	48.22476657	98.98292358
50000	94.23203546	35.26882999	98.94734536
100000	97.64946636	57.35461234	99.28746719

The results of this test corroborate that the HiCuts and Bit Vector are slower than DimCut. Both algorithms show the trend is more or less linear on the number of rules up to 10000 rules.

Through a series of experiments, we found that the algorithm reliably exhibits improved performance and accessibility with the proposed parameters and formula setting. The evaluation results are regulated in a directly analogous manner. As most packet classification algorithms are based on heuristics, different rule sets with different structures and sizes are inclined to offer very diverse results.

According to the data analysis and graphs, it is, therefore, substantiated that the proposed algorithms, based on decision tree, make packet classification faster, as compared to HiCuts, HyperCuts and BV algorithms. The HyperCuts has higher performance rather than the HiCuts and BV and the HiCuts is acting better rather than BV.

V. CONCLUSION

This paper aims at the evaluation concerns for high performance packet classification algorithms, which is a vital feature in Firewalls, routers, network security and quality of service (QoS) assurance. To accomplish a reliable performance, an algorithm should be conceived to blend the best characteristics of all approaches, besides optimizing the time-space transaction.

Our work's main contribution rests in the comprehensive and uniform appraisal of HiCuts, HyperCuts, Bit Vector 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, Number of Search, Rule Memory access, Depth of the tree structure and Threshold. Each test has been repeated three times to work out the average amount for the final results.

Multi-threading is a well-known programming and execution model that permits multiple threads to exist within the setting of a single process. We utilized multi-threading by using OpenMP to implement the DimCut and achieved better performance and results.

The concluding findings and contributions have been illustrated in graphics for operational demonstration. We opine that DimCut can be a viable Packet Classification algorithm that delivers a robust execution, besides allowing room for system designers to substitute components, and as a result aid the research and design community overall.

Further studies would, therefore, be necessitated to delve into more orderly systems for honing the configurable parameters, in order to upgrade the adaptive decision-tree construction procedures and rule set structure.

REFERENCES

[1] Singh, S., Baboescu, F., Varghese, G., & Wang, J. (2003). 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.

[2] Gupta, P., & McKeown, N. (1999). Packet Classification Using Hierarchical Intelligent Cuttings. In Proceedings of IEEE Symp. High Performance Interconnects (HotI), 7.

[3] Vamanan, B., Voskuilen, G., & Vijaykumar, T.N. (2010). EffiCuts: optimizing packet classification for memory and throughput. In Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, New Delhi, India.

[4] Sundström, M. (2007). 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.

[5] Amirjahanshahi, H., Poustchi, M., & Acharya, H. (2013). 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.

[6] OpenMP Architecture Review Board, (2013). Available at http://en.wikipedia.org/wiki/OpenMP and http://openmp.org/wp/.

[7] Taylor, D. (2005). Survey & Taxonomy of Packet Classification Techniques. In Proceedings of ACM Computing Surveys (CSUR), vol 37, Issue 3, pp. 238 - 275, September.

[8] Song, H., & Turner, J. (2011). Toward Advocacy-Free Evaluation of Packet Classification Algorithms. In IEEE Transactions on Computers, vol. 60, MAY.

[9] Srinivasan, V., Varghese, G., Suri, S., & Waldvogel, M. (1998) .Fast and scalable layer four switching. In Proceedings of ACM Sigcomm '98, pp. 191-202, Vancouver, Canada.

[10] Waldvogel, M., Varghese, G., Turner, J., & Plattner, B. (1997) .Scalable High Speed IP Routing Lookups. In Proceedings of the ACM SIGCOMM, 25-38.

[11] Srinivasan, V., & Varghese, G. (1999). 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.

[12] Gupta, P., & McKeown, N. (1999). 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.

[13] Feldmann, A., & Muthukrishnan, S. (2000). 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.

[14] Stihdis, D., & Lakslunan, T.V. (1998). 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.

[15] Qi, Y., & Li, J. (2006). 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.

[16] Song, H., Turner J., & Dharmapurikar, S. (2006). Packet Classification Using Coarse-Grained Tuple Spaces. In Proceedings of the ACM/IEEE Symp, Architecture for Networking and Comm., Systems (ANCS ’06), 41- 50.

[17] Srinivasan, V., Suri, S., & Varghese, G. (1999). Packet Classification Using Tuple Space Search. In Proceedings of the ACM SIGCOMM, citeseer.ist.psu.edu/srinivasan99packet.html.

[18] Baboescu, F., & Varghese, G. (2001). Scalable Packet Classification. In Proceedings of the ACM SIGCOMM.

[19] Abdelghani, M., Sezer, S., Garcia, E., & Jun, M. (2005). Packet Classification Using Adaptive Rules Cutting (ARC). In Proceedings of the IEEE Telecommunications, advanced industrial conference on telecommunications/service assurance on telecommunications workshop.

[20] Amirjahanshahi, H., Poustchi, M., & Acharya, H. (2014). Modification on Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut) with Analysis. In proceeding of the Research journal of Recent Sciences, Vol. 3, issue 8, August issue (in-press).