A STUDY & COMPARATIVE EVALUATION OF PACKET CLASSIFICATION ALGORITHMS

HEDIYEH AMIRJAHANSHAHI SISTANI

Department of Computer Studies and Research, Symbiosis International University, Pune, India
Email: hediehamirjahanshahi@yahoo.com

Abstract: In a field of incessantly expanding Internet connectivity, along with intensifying computer security threats, protection-conscious network applications technology is growing in prominence. Packet classifiers are exhaustively engaged for various network applications in distinctive types of network devices such as Firewalls and Router, besides many more. Comprehending the actual accomplishment of endorsed packet classifiers is essential for both algorithm designers as well as clients. Nonetheless, this is sometimes tedious to undertake. Each pioneering algorithm put out is evaluated from different views and is initiated on diverse theories. Bereft of a common establishment, it is fundamentally impossible to compare different algorithms in a direct fashion. This, in a way, helps the system facilitators to smoothly select the most appropriate algorithm for their actual applications.

Selecting a feeble algorithm for an application can lead to major expenses, particularly in the case of packet classification in network routers, as packet classification is essentially a tough problem and all contemporary algorithms are created on explicit heuristics and filter set features. The performance of the packet classification subsystem is of paramount importance for the collective success of the network routers.

In this study, we have conducted an advanced investigation of the existing algorithms to offer a comparative evaluation of a number of known classification algorithms that have been deliberated for both software and hardware applications. We have also explained our previously proposed DimCut packet classification algorithm, and associated it with the TSS, BV, HiCuts, HyperCuts and Woo packet classification algorithms with the comparative evaluation analysis. This comparison has been implemented on applications created on the matching principles and design selections from diverse sources. Performance measurements have been obtained by feeding the implemented classifiers with a large number of random Rules and Packets in an identical test scenario.

Keywords: Firewalls; Packet Classification; TSS; Bit Vector; HyperCuts; HiCuts; DimCut

I. INTRODUCTION

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. 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.

In this study, we have elucidated our erstwhile recommended DimCut packet classification algorithm, and compared DimCut with the TSS, HiCuts, HyperCuts, Woo 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 TSS, BV, HiCuts, HyperCuts and Woo packet classification 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.

II. RELATED WORK

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, when the number of rules increases, the result is inadequate, either towards search time or memory usage.

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. 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.

Tuple Space Search (TSS) is based on filter set grouping, where filters in a set are rearranged into separate subsets with definite common features.

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.

III. TSS, BV, HICUTS, HYPERCUTS, WOO, AND DIMCUT ALGORITHMS

A. TSS

In the tuple space search algorithm (TSS), the rules groups into a set of tuples according to their prefix lengths specified for different header fields then each group stores in a hash table. The memory consumption and the search time are depending on the number of tuples.

A tuple is defined as a vector of m lengths, where m is the number of fields in a rule. For example, in a 5-field rule set, the tuple means the length of the source IP address prefix is 5, the length of the destination IP address prefix is 7, the length of the protocol prefix is 6 (an exact value), the length of the source port prefix is 0 ("don't care"), and the length of the destination port prefix is 8 (an exact value). All rules should partition into the different tuple groups, and then should perform packet classification across all the tuples to find the best matched rule. Consider the example that shown in the following table I. There are 5 rules; each rule has only 3 fields. The tuples build according the fields and each rule has its tuple specification. Table II is shown the rules partitioning into tuple groups.

Table I: Shows the 5 rules with only 3 fields and their tuple specification.

Rule ID	Field 1	Field 2	Field 3	Tuple
1	11*	111*	10*	[2, 3, 2]
2	101*	110*	11*	[3, 3, 2]
3	10*	100*	10*	[2, 3, 2]
4	100*	111*	11*	[3, 3, 2]
5	10*	11*	101*	[2, 2, 3]

Table II: Shows the rules partitioning into tuple groups.

Tuple ID	Tuples	Rules ID
1	[2, 3, 2]	1, 3
2	[3, 3, 2]	2, 4
3	[2, 2, 3]	5

During packet classification, the hash tables will search to find the exact match among specific tuple and rules. There is a simple optimization, called tuple pruning, which can decrease the number of hash tables queried per lookup and reduce the number of tuples that has to be searched during the packet classification. As basically the number of unique prefixes matched on a particular field is smaller, the tuple pruning performs single field lookups by longest prefix match (LPM) to determine a subset of tuples for which there are matching rules. While using these optimizations, the other problem can appear from the other side, which is a tradeoff between the storage and performance. The basic problem with the tuple space search algorithm is that the number of tuples is too large.

B. Bit Vector

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 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. To implement the BV, we used the binary search technique to build the single field lookup data structure for retrieving the bit vectors.

C. HiCuts

The packet classification algorithm, Hierarchical Intelligent Cuttings proposed by Gupta and McKeown. 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.

spmf * number of rules at node r $\geq \sum$ 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.

D. HyperCuts

HyperCuts is a decision tree-based packet classification algorithm similar to HiCuts with some differences. HyperCuts allows cutting on multiple dimensions rather than only one dimension in HiCuts, and also it presents some additional optimization improvements such as: node merging, rule overlap, region compaction, pushing common rule subsets upwards.

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.

E. WOO

The Woo packet classification algorithm is a decision tree-based classification algorithm with a more direct view over the rule set. It is a modular composition of two procedures: the first to decompose large rule set into small rule buckets of a fixed maximum size and the second to process rule buckets of limited size to find a match. The framework contains three stages: an index jump table, search trees, and rule buckets. The bits used for the jump table address are selected such that every rule specifies those bits. When rule contain "don't cares" in jump table address bits, it must be stored in all search trees associated with the addresses covered by the jump index. For each entry in the index jump table that is addressed by at least one rule, a search tree is constructed [19].

The algorithm selects the preferences bits to set the bit string, by bit j definition as:

Preference [j] = $(D_j - D_{min}) / (D_{max} - D_{min}) + (N^*_j - N^*_{min}) / (N^*_{max} - N^*_{min})$ where, $D_j = | N0_j - N1_j |$ and $N0_j$, $N1_j$ and N^*_j are the number of 0, 1 and * at bit position j.

The rule bucket depth can be used as an efficient tunable control to reduce memory usage at the expense of increased search time.

F. 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.

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. 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 = $495.22 + (.034 * N) + (9 * 10^{-7} * N^2) + (6.23*10^{-12} *N^3)$, 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.

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.

IV. 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… 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 Figure 1, the blueprint of the simulated experiment is shown.

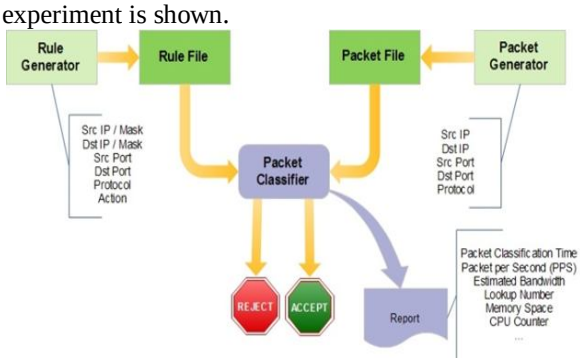

Figure 1: Simulated experimented methodology model setup.

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.

V. EXPERIMENTAL RESULTS

The worst case scenario is provided for tests and tests have been conducted multiple times to calculate the average.The following graphs display evaluation result of the BV, TSS, HiCuts, HyperCuts, Woo and DimCut packet classification algorithms.

Figure 2 presents that, while the rules are increasing the rule classification time also increases. The both DimCut and HyperCut act better with respect to time consumption and DimCut is faster than the others during the progress of rule classification process. The HiCut and BV rule classification time consumption is more reasonable than the TSS and Woo. Figure 3 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 and TSS have the worst action.

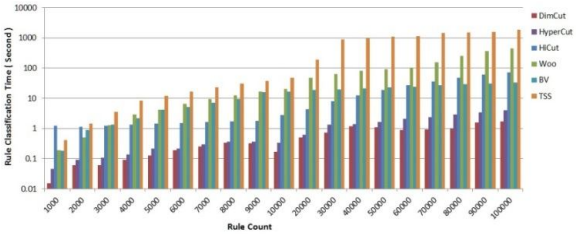

Figure 2: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure the rule classification time (second).

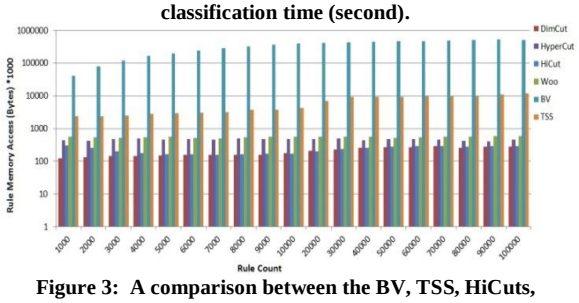

Figure 3: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure the rule memory access (Byte).

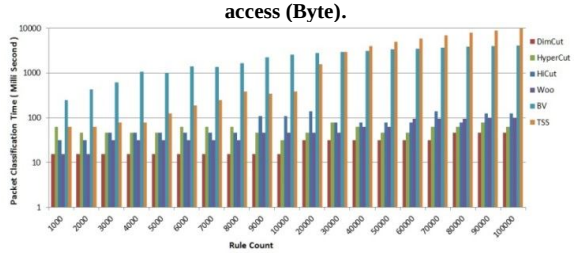

Figure 4: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure the packet classification time (Milli second).

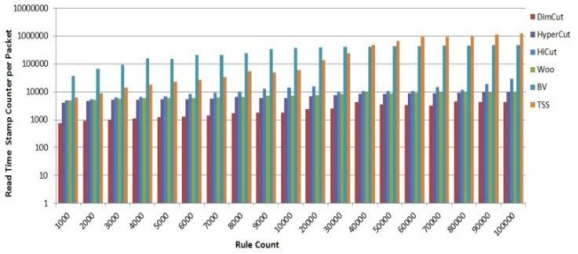

Figure 5: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure the total number of time stamp counter per packet.

Figure 4 shows when the rules increase the packet classification time also increases, but the DimCut is acting better with respect to time consumption. DimCut is faster than other ones; the BV and TSS are slower than the others during packet classification action. Figure 5 shows CPU time stamp counter per packet during classification process, which counts the number of cycles for each packet during the packet classification.

Figure 6 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 and Woo are acceptable but the BV and TSS are the worst action. DimCut has 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 Woo and Bit Vector algorithms when compared to HiCut, HyperCuts, TSS and DimCut algorithms, as can be seen from Figure 7. 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.

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 in TSS also is dramatically decreased for more than 20000 rules.

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, 99.2% over the Bit Vector, 99.9 over the Woo, 80.2% over the

HyperCuts and 99.9% over the TSS, as can be seen from Table III and IV. Table V and VI showthe DimCut can provide an improvement of up to 45.5% in Number of Rule Search calculation over the HiCuts, 70.9% over the HyperCut, 77.7% over the Woo, 99.9% over the BV and 99.9% over the TSS for the given rules.

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.

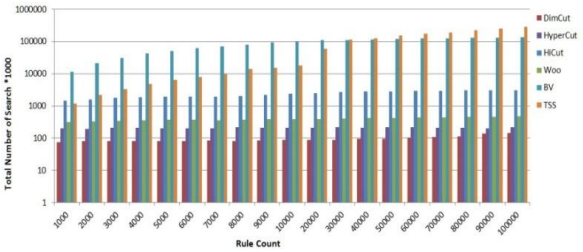

Figure 6: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure the total number of search.

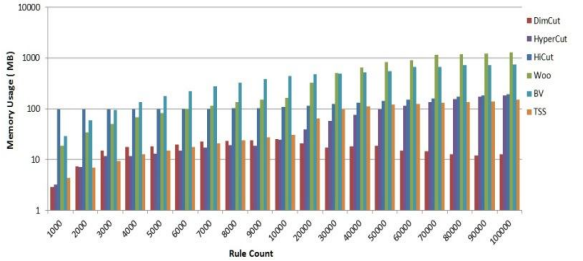

Figure 7: A comparison between the BV, TSS, HiCuts, HyperCuts, Woo and DimCut, to measure of memory usage (M).

Table III. The percentage of improvement of DimCut rather than HiCut, HyperCut for the given Rules in worst case scenario.

Rule Counts	Read Time Stamp Counter per Rule	
	HiCut	HyperCut
1000	99.13	80.25
5000	91.10	42.00
10000	91.38	48.22
50000	94.23	35.26
100000	97.64	57.35

Table IV. The percentage of improvement of DimCut rather than Woo, BV and TSS for the given Rules in worst case scenario.

Rule Counts	Read Time Stamp Counter per Rule		
	Woo	Bit Vector	TSS
1000	91.83	94.29	96.58
5000	97.31	96.91	99.05
10000	99.07	98.98	99.58
50000	99.85	98.94	99.82
100000	99.94	99.28	99.93

Table V. The percentage of improvement of DimCut rather than HiCut, HyperCut for the given Rules in worst case scenario.

Rule Counts	Number of Rule Search	
	HiCut	HyperCut
1000	45.58	70.92
5000	23.26	61.94
10000	15.96	62.88
50000	33.33	58.06
100000	45.45	24.56

Table VI. The percentage of improvement of DimCut rather than Woo, BV and TSS for the given Rules in worst case scenario.

Rule Counts	Number of Rule Search		
	Woo	Bit Vector	TSS
1000	77.73	99.69	97.07
5000	73.26	99.92	99.39
10000	69.75	99.57	99.76
50000	56.12	99.88	99.96
100000	29.96	99.89	99.94

CONCLUSION

This paper endeavors at the evaluation matters for high performance packet classification algorithms, which is a critical attribute in Firewalls, routers, network security and quality of service (QoS) assurance. To carry out a trustworthy implementation, an algorithm should be created to combine the superlative features of all methodologies, in addition to augmenting the time-space operation.

Our design's key input lies in the wide-ranging and homogeneous assessment of TSS, HiCuts, HyperCuts, Woo, Bit Vector and DimCut classification algorithms that have been executed with generic standards and appraised in a common test stratum, 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 conceive the mean quantity for the end result.

Supplementary research would, for that reason, be required to investigate into more compliant systems for perfecting the design factors, so as to reconstruct the variable decision-tree construction procedures and rule set configuration.

REFERENCES

1. S.Singh, F. Baboescu, G. Varghese &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.
2. P. Gupta & N. McKeown, "Packet Classification Using Hierarchical Intelligent Cuttings", in Proceedings of IEEE Symp. High Performance Interconnects (HotI), 7, 1999.
3. 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.
4. 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.
5. H.Amirjahanshahi, M. Poustchi &H. Acharya, "Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut)", the Research journal of Recent Sciences, 2(8), pp.31-39, August, 2013.
6. D. Taylor, " Survey & Taxonomy of Packet Classification Techniques", in Proceedings of ACM Computing Surveys (CSUR), vol. 37, Issue 3, pp. 238 - 275, September, 2005.
7. H. Song & J. Turner, "Toward Advocacy-Free Evaluation of Packet Classification Algorithms", in IEEE Transactions on Computers, vol. 60, MAY, 2011.
8. V. Srinivasan, G. Varghese, S. Suri&M. Waldvogel, "Fast and scalable layer four switching", in Proceedings of ACM Sigcomm '98, pp. 191-202, Vancouver, Canada, 1998.
9. M. Waldvogel, G. Varghese, J. Turner &B. Plattner, "Scalable High Speed IP Routing Lookups", in Proceedings of the ACM SIGCOMM, 25-38, 1997.
10. V. Srinivasan &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.
11. P. Gupta &N. McKeown, N, "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.
12. A. Feldmann&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.
13. 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.
14. Y. Qi&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.
15. H. Song, J. Turner &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.
16. V. Srinivasan, S. Suri&G. Varghese," Packet Classification Using Tuple Space Search", in Proceedings of the ACM SIGCOMM, citeseer.ist.psu.edu/srinivasan99packet.html, 1999.
17. F. Baboescu&G. Varghese, "Scalable Packet Classification, in Proceedings of the ACM SIGCOMM, 2001.
18. M. Abdelghani, S. Sezer, E. Garcia &M. Jun, "Packet Classification Using Adaptive Rules Cutting (ARC)", in Proceedings of the IEEE Telecommunications, advanced industrial conference on telecommunications/service assurance on telecommunications workshop, 2005.
19. 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, 2000.
20. H.Amirjahanshahi, M. Poustchi &H. Acharya, "Modification on Packet Classification Algorithm Based on Geometric Tree by using Recursive Dimensional Cutting (DimCut) with Analysis", the Research journal of Recent Sciences, Vol. 3, issue 8, August issue (in-press), 2014.

★★★