Three Different Designs for Packet Classification

HATAM ABDOLI
Computer Department
Bu-Ali Sina University
Shahid Fahmideh street, Hamadan
IRAN
abdoli@basu.ac.ir http://www.profs.basu.ac.ir/abdoli

Abstract: If we analyze real life filter sets (classifiers) and also packet classification requirements, it seems that distribution of rule scope is non-uniform and in some sub spaces is denser inside the total space of classifiers. These features guided us to add "cut point heuristic" to HiCuts, one of the most efficient algorithms and resulted in two new optimized designs for HiCuts, named B-HiCuts and Hist. Also one of the hardware based and fast solution for classification is use of TCAM memory to implement classifiers, but TCAMs are expensive and are not efficient for range fields. The third approach in this paper introduces a hybrid scheme to do a part of search on prefix fields in TCAM and then to check other fields, the search process will be traversed to RAM section in the second level. Synthetic classifiers are made by ClassBench and used to simulate and evaluate performance of proposed designs. The most specifications of proposed methods are balancing of decision trees and reducing the consumed memory for B-HiCuts and Hist and also solving range to prefix conversion and multi match classification problems in the last proposed design.

Key-Words: - Balanced tree, HiCuts, Heuristic, Packet classification, Packet filter, Router, TCAM.

1 Introduction

Traditional Internet routers just offered one type of service in which packets with the same destination was served identically in a first-come first-served manner. Modern routers because of quality of service for different applications should support mechanisms for services such as admission control, traffic shaping, security, resource allocation, queuing [1]. Providing all these requirements is feasible if routers are capable to classify different traffic flows base on several fields in packet's header, called packet classification. We can say that packet classification is a multi-dimensional form of IP lookup and finding longest prefix matching, to provide next-hop in routers.

1.1 Definition of Packet Classification Problem

The ability to classify each incoming packet is called packet classification and is based on an arbitrary number of packet header fields. The role of packet classification is important in special services such as VPNs, firewalls and differentiated services, and influence wire-speed routing. Since the packet classification problem is naturally difficult and is very complicated in the worst case of search time or the consumed memory, most of the proposed

Providing different levels of services by ISPs such as QOS [2], has guided us to a new generation of routers, named flow-aware routers in which agreements between users and ISPs express in terms of rules on incoming packets. The router's rule set is called policy database or classifier. Each rule specifies a specific information classification or flow that every incoming packet may belong to it based on its header field. For example all packets with the same destination IP address may define a flow in the classifier. The classifier may produce by network administrator or automatically [**3**].
According to each flow, an action (or actions) is specified on incoming packet. The incoming packet may match with more than one rule. Packet classification is finding the highest priority matched rule in classifier that specifies the best action on the incoming packet [1].

1.2 Previous Works

Algorithms have some weaknesses and limitations. Besides being fast and memory efficient, algorithms have to possibly be able to implement in software or hardware and also be scalable in the number of rules and dimensions (number of fields).
The simplest usable structure for this problem is a linked list of rules which have been sorted on priority. Each incoming packet is sequentially compared with the rules in the list until the first rule

matching on the fields of packet. Although this method is easy and makes optimum use of memory, has in the worst case, the lowest speed of classification.

The fastest and simplest possible solution for packet classification is the use of ternary content addressable memory or TCAM which its complexity of consumed memory equals to number of rules. These associative memories in addition to zero and one are capable to store don't care bits. Yet, this method has some limitations such as low scalability, high price, high power consumption, and static structure. Some optimized algorithms based on TCAMs have recently been designed, decreasing the cost of converting range into prefix and solving multi match classification [4], however the above limitations haven't solve perfectly. Another approach proposed in [5] to support IPv6 packets, but its structure is very complex and expensive and the algorithm has been evaluated by only small classifiers.

Some of algorithms have been designed for classification in two dimensions or have enough efficiency in this case such as Area-based Quad Tree (AQT)[6], and grid-of-tries [7]. An improved version of tuple space search has been proposed to make faster search and update for two-dimensional classification. But this approach is not scalable to more dimensions and also is not practical for IPv6 packets, having 128 bit length IP addresses [8].

A class of algorithms focuses on the specifications of classifiers resulting to heuristics to solve problem as a specific case and with the lowest possible cost. RFC and HiCuts belong to this class of algorithms which support the required speed [1]. But in contrast with HiCuts, RFC is costly in consumed memory and is not scalable with number of classifier rules. HSM which is a changed kind of RFC, in spite of the low consumption of memory, has more search time in average case and worst case compared with RFC [9]. Cross products [7] and algorithm of Baboescu and Varghese [10], also are examples of heuristic algorithms, but all of them have low scalability or weak behavior in the worst case.

BDDs structure [11] is a fast hardware solution, but the preprocessing time to build its static structure is very complicated and time consuming in the worst case.

HiCuts is one of the most efficient algorithms trying to decrease the search time by making cuts in geometric space of the problem. Data structure of HiCuts is a decision tree which is constructed based on rules of classifier in the preprocessing stage. Internal nodes distribute the search space while leaves contain a limited number of rules sequentially. So, classification of packet can be done by traversing the tree and a linear search in the leaf [1]. The number of cuts for each dimension (nc) is determined by the heuristic which can balance the trade off between depth of the tree (search time) and the consumed memory. The cut dimension is chosen in a way to minimize finally, maximum number of colliding rule set of the cuts. Algorithm can tune the number of cuts and consequently, the consumed memory with two parameters in a function. One of them is binth, to set maximum number of rules that can be searched sequentially in leaves, and the other, spfac is a factor, specifying the amount of consumed memory resulted from the cuts.

HyperCuts, A modified version of HiCuts, allows more than one dimension to be cut in each node to decrease occupied memory space and memory accesses [12]. Some solutions show good results in average case such as Cheng algorithm that uses cache and interpreting techniques [13].

2. proposed designs

Having mentioned all classification algorithms, it is clear that HiCuts with specifications such as scalability, low memory consumption and reasonable speed, is one of the most efficient. But choosing suitable point for making cut in the intended dimension is what that the algorithm designers haven't worked on. In HiCuts by using a heuristic, number of cuts (Nump) and using another, cut dimension are specified and then this dimension will be divided into equal pieces. Using this method in HiCuts, in the worst case, when the density of rules scope is considerable asymmetric and has non-uniform distribution or in other words the density of rules in special subspaces is more than other places, causes the HiCuts tree to grow in a unbalanced manner, in proportion to rules density. This causes the depth of tree and search time to increase in more dense subspaces and also extreme increasing of the consumed memory. It is obvious that the increasing of the tree's maximum depth, in the worst case, resulting to an inefficient hardware pipelined implementation [14].

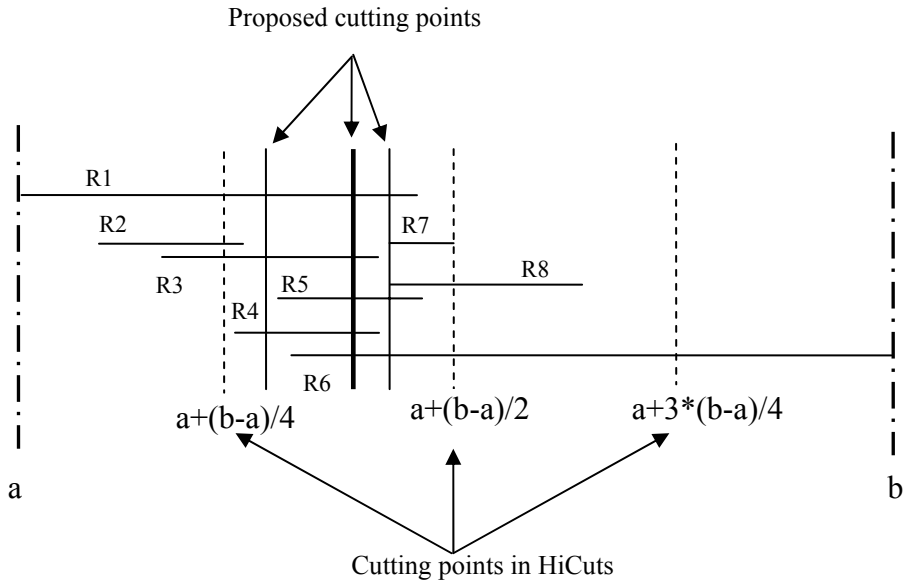

Fig.1 example of choosing 3 cut points in one dimension for rules corresponding to a node. Cuts in HiCuts are equal but suggested cuts are balanced in their colliding rule set.

This happens while classifiers regarding the uses caused the problem of packet classification, have basically a non-uniform distribution in the fields of their own rules scope [15]. So, HiCuts displays a weak behavior in this condition while the results of simulations also confirm this claim.

2.1 B-HiCuts[1]

According to the previous points, besides the mentioned heuristics for HiCuts, we can add a new heuristic which specifies cut points to balance the HiCuts like a multi-way B-tree [16]. This heuristic is defined as follows:

Cutting points Heuristic: It's a heuristic that the preprocessing algorithm uses to determine the points of cuts in a node and consequently its decision tree will be balanced. For this purpose, we can choose some points for the cut so that the number of rules in colliding rule set for each piece, resulted from the cut, distribute equally or with the least possible difference [14]. Colliding rule set for each piece are the ones which have at least one common point with that piece.

For example, consider fig.1 that indicates one of the dimensions of a typical node consisting of 8 rules in range [a, b]. Suppose we want to divide the node into 4 pieces by heuristic of choosing the number of cuts. Three cut points of HiCuts for cutting the range into equal pieces are shown in the figure with dotted line. The pieces resulted from the cuts have the colliding rule set from left to right: {R1, R2, R3}, {R1, R2, R3, R4, R5, R6, R7, R8}, {R6, R8} and {R6}. As you can see the number of rules in the colliding rule sets, they are distributed with no balancing, so that the second piece includes all the rules of the node. The result of this cutting is unbalanced growth of the equivalent decision tree and increasing of its maximum depth.

The proposed cut points in this figure are chosen by vertical dashed lines. The boldest line between the other two vertical lines specifies the first cut point. Based on the proposed method, this point is chosen in a way that the number of colliding rules in two resulted pieces be closed enough to each other and to minimize their maximum, among the examined points. After this stage two other cut points are chosen in two resulted ranges from the first cut point and by the same method. If the number of cuts be more than 8, we can repeat the above stages hierarchically for each piece.

If the number of cuts aren't power of 2 (for example k), we can design the algorithm in a way that it cuts the node all at once into intended rules while the colliding rules of each part gets close to n/k, which n is the total number of colliding rules in the cut dimension for the node. The colliding rule set for 4 pieces resulted from the proposed cut points of fig.1, from left to right, are: {R1, R2, R3, R4} and {R1, R3, R4, R5, R6} and {R1, R3, R4, R5, R6} and {R1, R5, R6, R7, R8}. The new sets have nearly the same number of rules and the biggest set consists of

[1] Balanced-HiCuts

5 rules. The result of choosing these cut points is a balanced decision tree. For finding these suggested cut points we can use different algorithms. For example, one way is a binary search in the specified range from the cut dimension. The other better way is examination of the start and end points of each range for every rule in the intended dimension [14].

2.2 Hist Structure

Binary Hist[2] decision tree, which is the basic structure of multi way Hist decision tree, is similar to kd-tree structure. The structure of binary Hist algorithm is a binary balanced (or nearly balanced) tree in which every internal node (not leaf) is cut in a dimension chosen by the cut point heuristic and is divided into subtrees resulted from the cut. The cut point has chosen providing the number of colliding rules in both side of this point have a better equality after splitting.

The heuristic of determining the cut dimension includes finding a dimension in which the maximum number of rules after splitting becomes minimal in all dimensions. The structure of Hist algorithm with multi way decision tree is constructed based on binary Hist and by compressing the layers of binary tree. This is done by elimination of pointers in compressed-layers nodes, to decrease depth of the tree and number of memory accesses [14].

Table 1. A typical two dimensional classifier.

Rules	X	Y
R1	011	001
R2	01*	10*
R3	11*	0*
R4	11*	10*
R5	0*	0*
R6	*	000

For example table 1 is a two dimensional classifier and fig.2 shows construction of its Hist structure. Part (A) and (B) of fig.2 indicate X and Y dimension of classifier's rules and their cut points by dotted lines. Part (C) is Hist binary tree in which circles indicate internal nodes and their values are dimension and points of cutting. Rectangles are leaves that include matching rules based on priority. Dotted rectangles indicate colliding rule set for each internal node.

Since the Hist structure is a balanced binary tree, we can modify it by elimination of some pointers. Structure of Hist with multi-way decision tree is constructed on binary Hist by compressing its levels. Level compression in tree is in fact elimination of tree pointers corresponding to compressed levels, and consequently reducing the tree depth and number of memory accesses. Number of level compression (LC) can be a constant value or variable. In the new structure every node has 2^{LC} children, 2^{LC}-1 cutting points in multiple dimensions. In the last level we have our previous leaves or null nodes. For example the constant LC=3 means that three levels of binary Hist will be compressed and every node has 8 children and 2^3-1 or 7 cutting points.

Fig.3 indicates level compression for binary Hist of fig.2 and constructing multi-way Hist with LC=2. The dotted rectangle shows the compressed nodes, resulted in the root of multi-way tree. It is clear that this structure will be useful if its binary tree is balanced such a complete or nearly complete binary tree.

2.3 A hybrid scheme

Before describing the hybrid approach, it is necessary to dig deeper into TCAM and its use in packet classification.

When a search key is given to TCAM, it should be compared with all entries simultaneously and return back first match, so entries should be filled according to their priority. Nowadays, we have TCAMs with 133 million searches per second in 144 bit entries, while the total capacity can be extended to 128K entries. But using TCAMs in packet classification has some limitations such as high power consumption, expensive price, and Multi Match Classification problem in applications which several or all the matched rules should be specified, because normally TCAMs only return rule with the most priority. New approaches have been proposed to solve this problem [17].

[2] Hierarchical splitting tree

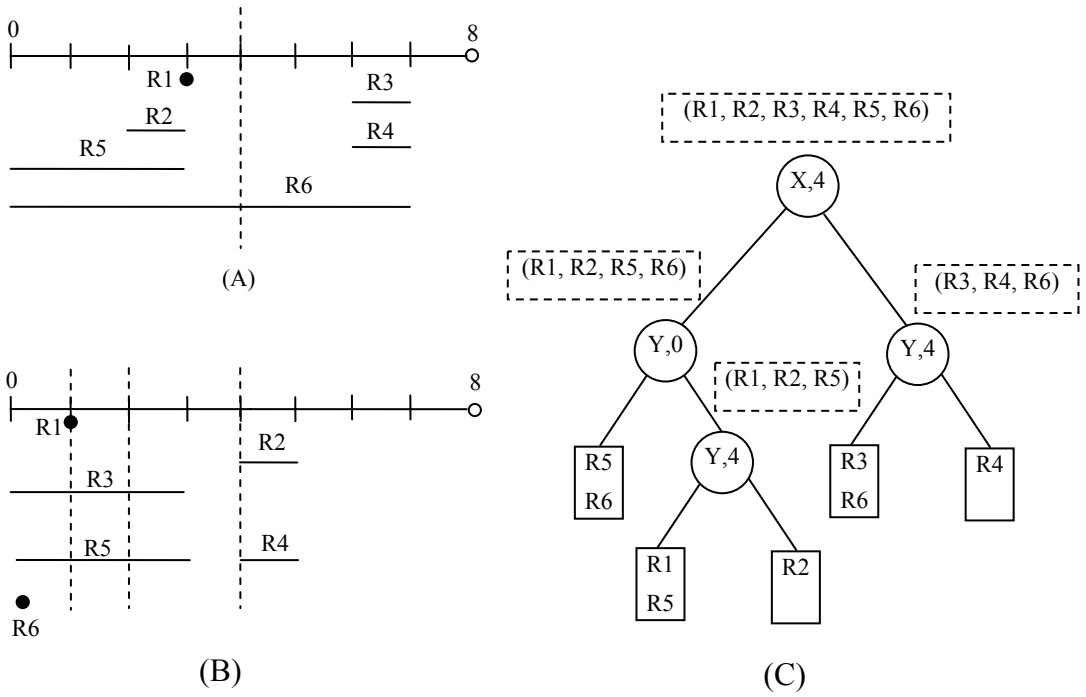

Fig.2 Hist tree for classifier of table 1. A) Rules and cut points in X dimension B) Rules and cut points in Y dimension C) Hist binary tree

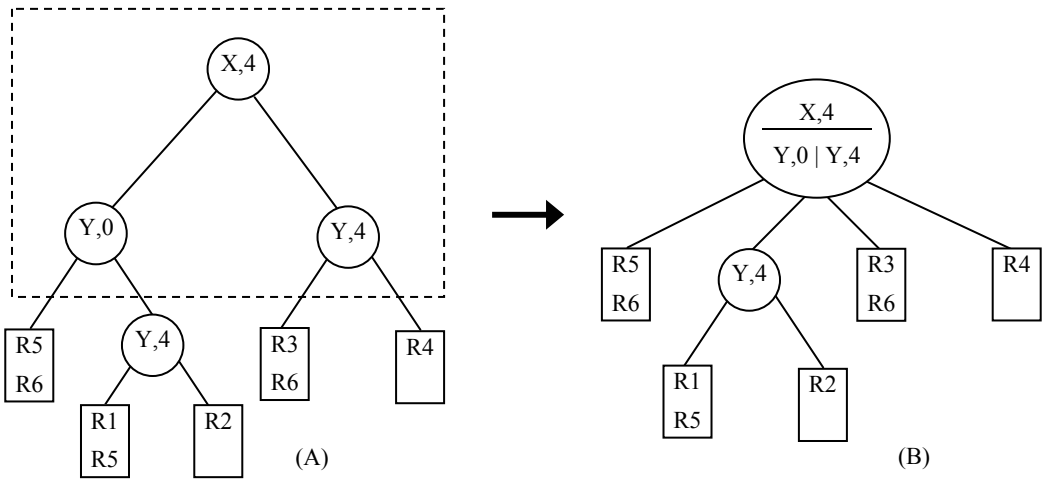

Fig.3 Level compression for binary Hist tree of fig.2. A) Binary Hist B) Multi-way Hist

The most important problem for TCAMs is rules with range fields should be converted to prefixes before storing in entries, resulting in increasing TCAM entries linearly, while percentage of range fields in classifiers are growing up[17]. In spite of some solutions [17, 18, 19], this problem yet has redundancy, reducing algorithm efficiency.

The basic idea in the hybrid scheme is using of TCAMs in part of matching which the fields are prefix type(such as source and destination IP address) and continuing the search for the remained fields in second level. So, we are not facing to converting of ranges to prefixes, which is the main problem of using TCAMs in classification. The global structure of proposed design is shown in fig.4. According to this approach, at first all the prefix fields of classifier will be searched in TCAM and based on the matched entry, a pointer will be used in the second level algorithm, to continue searching on the remained fields (non-prefix), resulting to find best matched rule.

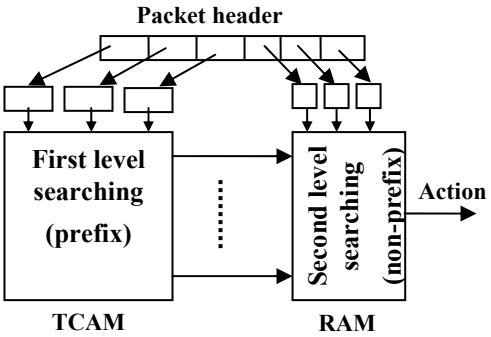

Fig.4 Proposed hybrid scheme for packet classification

Based on this model, if number of rules or fields increase in the future, we can support updates by reconstruction of decision trees in the second level according to new conditions. An interesting feature of this hybrid solution is flexibility to select all algorithms which use decision tree such as HiCuts, B-HiCuts or FIS structure. In this research we have selected B-HiCuts as the second level algorithm.

To know more details about the operation, a five dimensional classifier has been shown in table 2, having destination IP, source IP, source port, destination port, and protocol fields.

Table 2. A typical 5-dimensional classifier

Rule	Source IP	Dest. IP	Source Port	Dest. Port	Protocol	Action
R1	228.2.12.x	x.x.x.x	80	>1024	TCP	A1
R2	228.2.12.x	x.x.x.x	*	80	*	A2
R3	x.x.x.x	x.x.x.x	*	*	*	A5

Fig.5 shows the proposed hybrid scheme for classifier in table 2. all of the non-prefix fields F1, F2, and F5 of rules have stored in TCAM entries. Based on these fields of header, parallel search will be done in TCAM and the matched row with the highest priority will determine the pointer for next level of classification.

The pointer in the proposed design points to root of a B-HiCuts decision tree. There is a tradeoff between memory consumption and depth of tree by choosing proper children of nodes and colliding rule set. If searching in the average case is acceptable in our application, a bit (bit F in figure) can be used to specify the matched rules which can be determined only by F1,F2,F5 fields and are not depended to other fields. For example if F=0, it means end of classification and the other bits in RAM specify the appropriate action, else the classification should be continued in a decision tree which is pointed by other bits of RAM. In fig.5 a packet with source IP address 228.2.12.1 and destination IP 200.200.200.1 is pointing to a root of a tree which has been cut on field F4 and point 1024. because the value of field F4 of incoming packet is 100 and is less than 1024, so we traverse through upper branch and then we can continue classification process by a sequential search on rules R2 and R3 to find the most prior rule, which is R2 in this example. Notice that binth parameter is 2 in this example.

If we use TCAM with wider entry (for example the available 288 bit wide TCAMs), it is possible to put 128 bit wide IP addresses and extend the model to support IPv6.

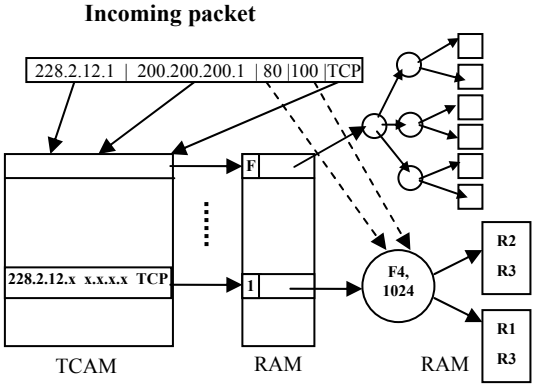

Fig.5. The proposed hybrid scheme for table 2. first part of search is in TCAM and it will continue in RAM on non-prefix fields by B-HiCuts.

3. PERFORMANCE EVALUATION

Since the proposed algorithms are based on HiCuts algorithm, we chose the simulation results for this algorithm as the criterion of comparison with our consequences. All synthetic classifiers used for evaluation of algorithms have five dimensions and each rule has five IPv4 fields including: source IP address, destination IP address, source port, destination port and type of protocol. The method of producing each rule of synthetic classifiers is based on the features in reference [15]. Values are used for ClassBench parameters to make a non-uniform synthetic classifier. So, 0.6 has been selected for skew and number of dimensions is five.

The hybrid design has been evaluated separately by comparing with HiCuts and B-HiCuts.

In this paper we just consider two more important criteria, search time and the consumed memory in algorithms. Simulations are carried out in C language on windows XP system.

3.1 Search Time Evaluation

The results of simulations show that for classifiers with non-uniform distribution, B-HiCuts and Hist have the lower depth comparing with HiCuts. That's because HiCuts has no heuristic for choosing a proper cut point and therefore, cuts go on blindly. Since the real search time is a function of memory access, therefore the depth of the tree is not a suitable criterion for the real search time. Results of simulation show that in classifiers with uniform distribution, the maximum number of access to Hist structure has no big difference with HiCuts structure. But on non-uniform classifiers, the suggested structures, especially Hist, in the worst case, have better efficiency in the number of memory accesses and consequently in the maximum of search time.

Fig.6 indicates maximum total number of memory access on classifiers with non-uniform distribution that simulated for band width equal to 4 bytes. To show the role of parameter "binth" (refer to section 1-2), two small classifiers have been simulated for binth=16 and for two bigger classifiers binth is 32.

In fig.7 search time of proposed hybrid algorithm has been compared with HiCuts in the number of memory access criteria. It is clear that search time of proposed algorithm is less than TCAM (which needs only one memory access) and is more than HiCuts. The overall access time is the numbers in fig.7 plus one access in TCAM.

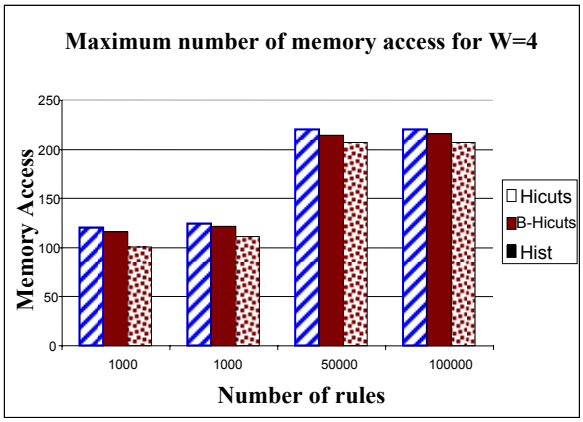

Fig.6 Maximum total number of memory access

3.2 Consumed Memory Evaluation

Comparing the results, shows that the structure of Hist has the lowest amount of consumed memory for both kinds of classifiers, especially on those with non-uniform distribution (see Fig.8).

This is because of the correct choice of cut points and the decreasing of internal nodes and leaves in Hist data structure.

B-HiCuts has generally less memory on the classifiers with non-uniform distribution than HiCuts [14]. Consider that the vertical axis of Fig.8, which shows the memory, has a logarithmic scale.

Memory consumption is shown in Fig.9 between hybrid scheme, HiCuts and TCAM. Because both TCAM and RAM memory have been used in the hybrid scheme, all of them have shown separately to make a better comparison for each type of memory.

Having shown in this figure, the consumed RAM in hybrid scheme is less than HiCuts and the consumed TCAM in hybrid design is also less than a completely implementation of packet classification in TCAM memory.

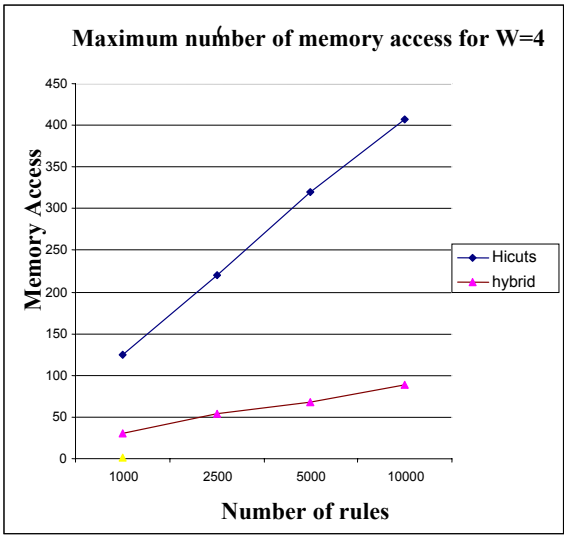

Fig 7. Maximum number of memory access for W=4 byte.

3.3 Evaluation of computational complexity

The complexity of computations is one of the criteria to evaluate efficiency of search algorithms in implementation phase. This criteria may include different parameters such as: arithmetic operations (multiplication, addition, …) logical operations (AND, OR, Shift, …), and conditional instructions. All of these can help us to choose a good platform in implementation, such as: hardware based, software based, FPGA based, ASIC based, parallel, or pipelined implementations. So, type and number of different processing is used in search operation in every algorithm are shown in table 3 and table 4, which specify above computations in internal nodes and leaves, respectively. W describes memory width in byte unit.

According to the tables, with an equal binth parameter, the amount of computations in leaves is equal in all algorithms. In Hist and B-HiCuts only memory access and conditional instructions are necessary.

Table3. Type and number of operations for search in internal nodes of algorithms.

algorithm / Type of processing	**Hist**	**B-Hicuts**	**Hicuts**
Number of memory accesses	$\lceil 22/W \rceil$	$\lceil (2+((\log SpltNo)+1)*4)/W \rceil$	$\lceil 14/W \rceil$
Addition & subtraction	—	—	2
If	5	(SpltNo) + 2	3
Division	—	—	2

Table 4. Type and number of operations for searching in leaves of algorithms.

algorithm / Type of processing	**Hist**	**B-Hicuts**	**Hicuts**
Number of memory accesses	(24*binth) /W	(24*binth) /W	(24*binth) /W
shift	4*binth	4*Binth	4*binth
If	binth	binth	binth

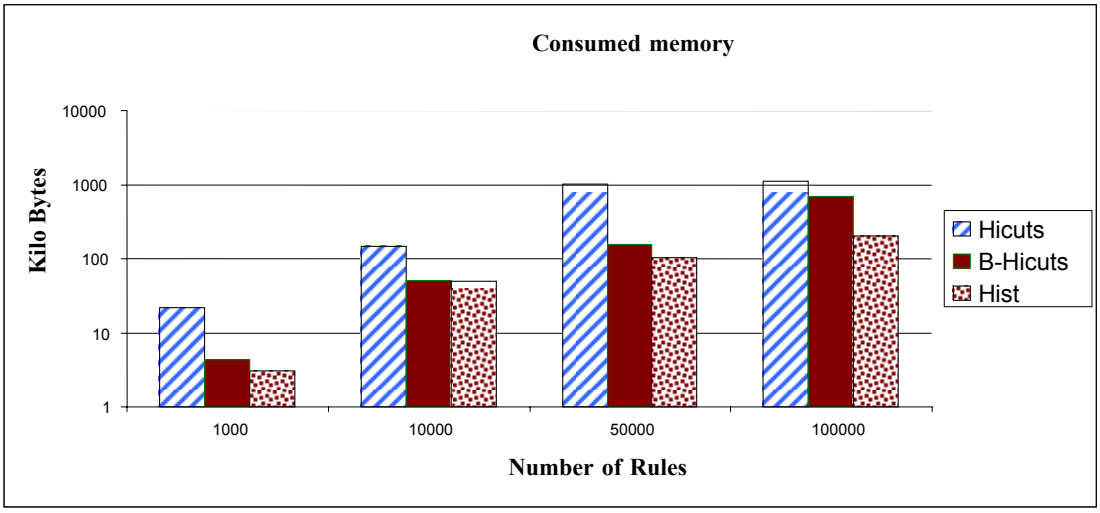

Fig.8. Consumed memory in three algorithms with binth=16. The vertical axis has logarithmic scale.

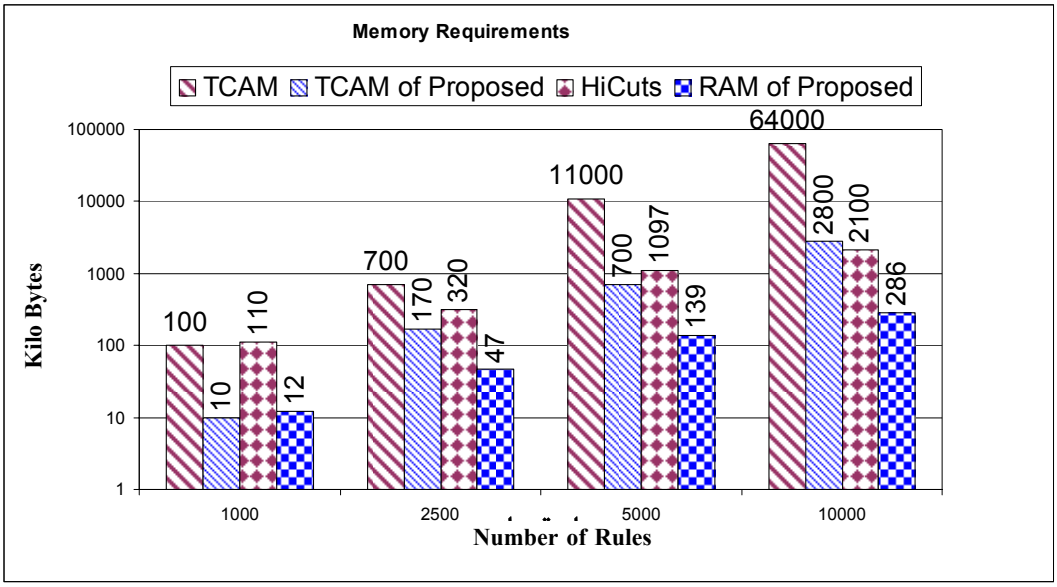

Fig. 9. A comparison between RAM of proposed hybrid scheme and RAM of HiCuts and also between TCAM of hybrid design and complete TCAM implementation.

4. CONCLUSION

Comparing the proposed algorithms indicates that HiCuts is one of the most efficient ones, but in the worst case and with real-life classifiers has an unbalanced decision tree leading to a large maximum depth. In this paper by defining the heuristic of cut point selection, two designs were suggested for improvement of HiCuts algorithm and balancing the decision tree. Since the Hist structure is binary and balanced, therefore we can eliminate the tree pointers and combine nodes of several levels as a compressed static structure which nodes have multi dimensional cut points in.

Proposed designs like HiCuts are extendable to IPv6. Because the trees of new designs are balanced and have a lower depth than HiCuts, their pipelined implementation, are more efficient and have lower delay. The main weakness of proposed approaches is long preprocessing (and update) time.

Also in this paper a new hybrid scheme has been defined based on using TCAM for prefix fields and another structure for other fields. Some specifications of this design according to simulation results are high speed, low complexity, convergence in producing data structures, acceptable memory consumption, and flexibility in choosing algorithms

in the second part of hybrid design. In the hybrid design the problem of converting range to prefix and also multi match classification problem have been solved and it can be easily extended to IPv6.

References:

[1] P. Gupta*, Algorithms for Routing Lookups and Packet* Classification*.* Ph.D. Thesis, Stanford University, December 2000.

[2] R. Chen, and Y. Tsai, and K.C. Yeh, and H.Y. Chen, Using policy-based MPLS management architecture to improve QOS on IP network*, WSEAS Transaction on computers*, Vol. 7, No. 5, 2008, pp. 341-350.

[3] H. Haitao and others, Traffic classification using en-semble learning and co-training, In WSEAS Proceedings of the 8th conference on Applied informatics and communications, Rhodes, Greece, 2008.

[4] L. Karthik, and R. Anand, and V. Srinivasan, Algorithms for advanced packet classification with ternary CAMs. *ACM Sigcomm*, 2005.

[5] X. Zhang, and B. Liu, IPv6-oriented 4*OC768 Packet Classification Scheme with Deriving-Merging Partition and Field-Variable Encoding Algorithm, *IEEE INFOCOM,* 2006.

[6] M. Buddhikot, and S. Suri, and M. Waldvogel, Space decomposition techniques for fast layer-4 switching. In *Proceedings of Protocols for High Speed Networks*, August 1999.

[7] V. Srinivasan, and S. Suri, and G. Varghese, Fast and scalable layer four switching. In *Proceedings of ACM Sigcomm*, September 1998.

[8] L. Xing, and J. Zhen-Zhen, and L. Wei-Chen, Hardware Based High Speed Two-dimensional Packet Classification, *WSEAS Transaction on computers*, Vol. 5, No. 2, 2006, pp. 301-305.

[9] B. Xu, D. Jiang, J. Li, HSM: A Fast Packet Classification Algorithm. In *Proceedings of IEEE 19th International Conference on Advanced Information Networking and Applications (AINA)*, Taiwan, 2005.

[10] F. Baboescu, and G. Varghese, Scalable packet classification. *ACM Sigcomm'01*, 2001.

[11] A. Prakash, and A. Aziz, OC-3072 Packet Classification Using BDDs and Pipelined SRAMs. In *Hot Interconnects*, Stanford University, CA, August 2001.

[12] F. Baboescu and G. Varghese, Packet Classification Using Multidimensional Cutting, In *Proceeding Of ACM SIGCOMM*, 2003.

[13] H. Cheng, Z. Chen, Scalable Packet Classification using interpreting a cross-platform multi-core solution, *13th ACM SIGPLAN* , 2008.

[14] H. Abdoli, *Review and design of packet classification algorithms*. MS thesis, Isfahan University of Technology, September 2003.

[15] D. E. Taylor and J. S. Turner, "ClassBench: a packet classification benchmark," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 499–511, 2007.

[16] A. Sleit, On using B-Tree for efficient processing for the boundary neighborhood problem*, WSEAS Transaction on systems*, Vol. 7, No. 7, 2008, pp. 711-720.

[17]Lakshminarayanan, K. and Rangarajan, A. and Venkatachary S., "Algorithms for advanced packet classification with Ternary CAMs", *In ACM SIGCOMM*, 2005.

[18] Spitznagel, E. and Taylor, D. and Turner, J., "Packet Classification Using Extended TCAMs", *In Proc. of ICNP*, 2003.

[19] Zheng, K. and Wang, Z. and Liu, B. "TCAM-based Distributed Parallel Packet Classification Algorithm with Range-Matching Solution", *in Proceedings of IEEE INFOCOM'2005*, March 2005.