Balanced HiCuts: an optimized Packet classification algorithm

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: - 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. After studying the characteristics of real life classifiers and also requirements of packet classification, it seems that distribution of rules scope is non-uniform and in some sub spaces have more density inside the total space of classifiers. This feature guided us to add "cut point heuristic" to HiCuts, one of the most efficient algorithms. Based on this new heuristic, two new optimized designs for HiCuts have been proposed and their performance are simulated and evaluated. The most specifications of proposed methods are balancing of decision trees and reducing the consumed memory.

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

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

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 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
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 [2], however the above limitations haven't solve perfectly. Another approach proposed in [3] 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)[4], and grid-of-tries [5]. A class of algorithms focus 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 [6]. Cross products [5] and algorithm of Baboescu and Varghese [7], also are examples of heuristic algorithms, but all of them have low scalability or weak behavior in the worst case. BDDs structure [8] 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 [9]. Some solutions show good results in average case such as Cheng algorithm that uses cache and interpreting techniques [10].

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 [11].

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 [12]. So, HiCuts displays a weak behavior in this condition while the results of simulations also confirm this claim.

2.1 B-HiCuts

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. This heuristic is defined as follows:

Cutting points Heuristic : It's a heuristic that the preprocessing algorithm uses to determine the

---

1 Balanced-HiCuts
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 [11]. 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 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 [11].

### 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

---

2 Hierarchical splitting tree
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 [11].

Table 1. A typical two dimensional classifier

<table>
<thead>
<tr>
<th>Rules</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>011</td>
<td>001</td>
</tr>
<tr>
<td>R2</td>
<td>01*</td>
<td>10*</td>
</tr>
<tr>
<td>R3</td>
<td>11*</td>
<td>0*</td>
</tr>
<tr>
<td>R4</td>
<td>11*</td>
<td>10*</td>
</tr>
<tr>
<td>R5</td>
<td>0*</td>
<td>0*</td>
</tr>
<tr>
<td>R6</td>
<td>*</td>
<td>000</td>
</tr>
</tbody>
</table>

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.

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 [12]. 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.

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. 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

Fig. 4 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.

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. 5).

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 [11]. Consider that the vertical axis of fig. 5, which shows the memory, has a logarithmic scale.

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.

Fig. 4 Maximum total number of memory access

Fig. 5 Consumed memory in three algorithms with binth=16. The vertical axis has logarithmic scale.

References: