International Journal of Power Control Signal and Computation (IJPCSC)
Vol. 5. No.1. pp.9-17,Jan-March 2013 ISSN: 0976-268X
www.ijcns.com

# An innovative approach of packet classification based on TCAM

[1]R.Jayanthi [2] J.Lakshmikanth [3] V.Geethapriya
[1]Sankara Deemed University, Kanchipuram, jayanthiravi96@gmail.com
[2]St Peter's University, Avadi, lakshmikanth_sai@yahoo.co.in
[3]St Peter's University, Avadi , vgeetha87@rediffmail.com

**Abstract**

**Multiple range of re encoding scheme have been proposed to enhance the effect of large range expansion and increase high capacity and large efficient power consumption and low heat generation of ternary content addressable memory based on the packet classification unfortunately we can have optimally solve the one dimensional problems and made complex interaction between dimensions and complicated multidimensional problems especially the total range of expansions required for each rule and each field is the product of range expansion. Therefore in this paper we have to be implemented multiple prefix alignment for solve the complicated multidimensional problem. This multidimensional prefix alignment problems potentially combination of the new techniques with TCAM optimization and re encoding scheme. The simulation result that our technique achieve at least 10 times more space reduction in case of TCAM space for an encoded classifier and its transformers that leads to improved throughput complexity and decreased power consumption.**

**Index Terms**

**Ternary content addressable memory, range reencoding,space reduction, classifier.**

## I. INTRODUCTION

Packet classification require a number of network services for instance routing, access-control in firewalls, policy based routing, provision of differentiated qualities of service, and traffic billing. In every case, it is required to decide the flow of arriving packet then conclude whether the packet is going to forward or filter. If it is forward, identify the set of service it should receive, or how much should be charged for transporting it. The tagging purpose is performed by a flow classifier (also called a packet classifier).Packet classifier maintains a set of rules, where all packet flow obeys at least one rule. The rules order Packet flow belongs to based on the packet header(s). For example, source and destination IP address values, and particular transport port numbers define the packet flow. Otherwise a flow could be basically defined by a destination prefix and a range of port values. As we shall see, in practice a number of different types of rules are used. This paper describes a technique for fast packet classification based on an about uninformed set of rules. We focus now only on the difficulty of identifying the class to which a packet belongs. The actions taken for each class (e.g. packet scheduling in an output queue, routing decisions, billing [2][3][4][8][11][12][13]) while attractive in its own right, is not the topic of this paper. The most eminent form of packet classification is used to route IP data grams. In this case, all of the packets intended to the set of addresses described by a common prefix may be considered to be part of the same flow. In the lead arrival to a router, each packet header is examined to conclude the Network-layer destination address, which identifies the flow to which the packet belongs.

Awaiting recently, longest-prefix matching for routing look ups could not be done at high speeds. Now that several fast routing look up algorithms have been developed, attention has turned to the wide-ranging problem of packet classification.

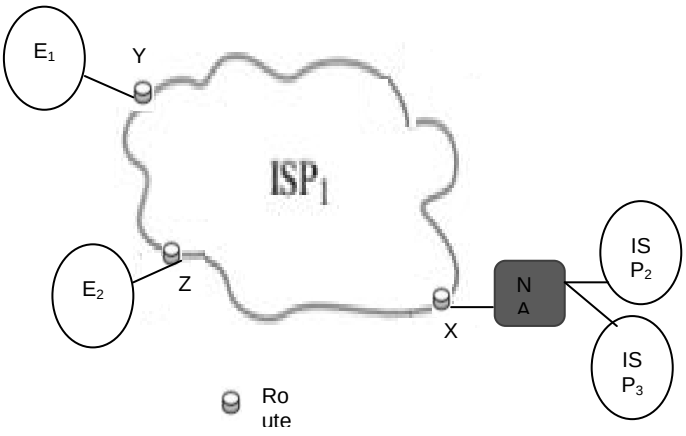



**Figure1: Example network of an ISP ( $ISP_1$) connected to two enterprise networks($E_1$ and $E_2$) and to two other ISP networks across NAP.**

Figure 1 shows $ISP_1$ connected to three different sites: two enterprise networks $E_1$ and $E_2$ and a Network Access Point (NAP) which is in turn joined to $ISP_2$ and $ISP_3$. $ISP_1$ provides a number of dissimilar services to its customers, as shown in Table 1.

| Service | Example |
|---|---|
| Packet Filtering | Den all traffic from $ISP_3$ (on interface X) destined to $E_2$ |
| Policy Routing | Send all voice-over-IP traffic arriving from $E_1$ (on interface Y) and destined to $E_2$ via a separate ATM network |
| Accounting & Billing | Treat all video traffic to $E_1$ (via interface Y) as highest priority and perform accounting for the traffic sent this way. |
| Traffic Rate Limiting | Ensure that $ISP_2$ does not inject more than 10Mbps of email traffic and 50Mbps of totaltraffic on interface X. |
| Traffic Shaping | Ensure that no more than 50Mbps of web traffic is injected into $ISP_2$ on interface X. |

Table 2 shows the categories or classes that the router at interface X should classify an incoming packet into. The classes specified may or may not be mutually exclusive must be noticed(i.e. they may be overlapping or non-overlapping), for example the first and second class in Table 2 overlap. When overlapping happens, we will go after the principle, in which rules closer to the top of the list take priority, with the default rule appearing last.

### 1.1 The Problem of Packet Classification

Packet classifier is used to perform the Packet classification, is furthermore called a policy database, flow classifier, or simply a classifier. A classifier is nothing a collection of rules or policies. A class specified by each rule that a packet may belong to based on some standard on F fields of the packet header, and connections with every class an identifier, classID. The action associated with the rule is uniquely specified by the this classID identifier . All rule has F components. The ith component of rule R, referred to as R[i], is a regular expression on the ith field of the packet header (in practice, the regular expression is restricted by syntax to a simple address/mask or operator/number(s) specification). A packet P is said to equal a particular rule R, if $\forall i$ , the ith field of the header of P satisfies the regular expression R[i]. It is suitable to think of a rule R as the set of all packet headers which could match R. When viewed in this method, two distinct rules are said to be either partially overlapping or non-overlapping, or that one is a subset of the other, with parallel set-related definitions. When two rules are not mutually exclusive will be assumed throughout in this paper.The tidy in which they appear in the classifier will determine their relative priority. We have to identify the whole forwarding table in a packet classifier is specified by , a packet classifier that performs longest-prefix address look ups, each destination prefix is a rule, the related next hop is its action, the pointer to the next-hop is the associated class ID. If we presume that the forwarding table has longer prefixes appearing before shorter ones, the look up is an example of the packet classification problem.

## II. RELATED WORK

Preceding work in optimizing T CAM-based packet classification systems plunge into three broad categories: circuit modification, classifier compression, and range re encoding.

### 2.1 Circuit Updation

Spitznagel et al. proposed adding comparators at each entry level to better accommodate range matching [14]. While this research direction is important, such solutions are hard to install due to high cost [6].

### 2.2 Compressed Classification

These optimizations convert a given packet classifier to another semantically equivalent classifier that requires fewer TCAM entries. The schemes in [10], [15], [25] focus on one-dimensional and two dimensional packet classifiers. The redundancy removal algorithms in [19] can shrink TCAM usage by eliminating all the redundant rules in a packet classifier. In [18], Dong et al. proposed schemes to reduce range expansion by frequently expanding or trimming ranges to prefix boundaries without changing the number of bits used to represent each dimension. They confirm correctness by using core effective region algorithms in [19]. In concentrate, they insert the new rule before the rule being modified and check if the new rule is redundant. In dissimilarity, our prefix alignment algorithm mitigates range expansion by intelligently adding bits

to a given dimension to increase the number of prefix boundaries. In [20] Meiners, et al. proposed finding locally minimal solutions by using greedy algorithm that along each field and combines these solutions into a smaller equivalent packet classifier. In [21], Meiners et al. proposed the first algorithm that can reduce a given classifier into a non-prefix ternary classifier.

**2.3 Re encoded of Ranges**

Prior range re encoding schemes falls into two categories:

i) Range re encoding that only consider rule set size, often at the expense of rule width [22], [23], [24], [25] and

ii) Range re encoding that attempt to both compress rule set size and rule width [26], [27], [28],[29]. In [24], Liu proposed a method that allocates specific TCAM column bits to represent ranges in a manner like to Lakshman and Stiliadis' software bitmap classification method [6]. Lakshminarayan et al. [30] proposed a scheme called fence encoding, which encodes interval ranges as a range of unary numbers. Fence encoding has an growth factor of one, meaning all ranges can be encoded with one string, but the number of unary bits required for each rule is affordable.

To reduce rule width, Lakshminarayan et,al. proposed DIRPE, which compresses the width of fence encodings at the expense of a larger expansion factor. Bremler- Barr and Hendler [22] proposed SRGE, which utilizes the structural properties of binary reflected gray codes to shrink range expansion without increasing rule width. Lunteren and Engbersen proposed a hierarchy of three methods, P 2 C, that can be used to compress both rule number and rule width . Two methods undertaking an expansion factor of one but have potentially larger rule widths. The third method has the best rule width compression at the cost of expansion factors larger than one. Bremmel-Bar et al. [26] purpose concrete algorithms for the P 2 C hierarchy. Pao et al. proposed a prefix inclusion method (PIC) that achieves better rule width compression than P 2 C [27], [28]. Che et al. [6], [23] and Pao et al. [27], [28] propose using TCAMs to re encode packets. software based packet classification using Reencoding. Lakshman and Stiliadis proposed to reencode each field's value into a bitmap that specifies the repression relationship among values and rules [30].

Given a reencoded packet, this method uses customized parallel AND gates to perform an connection of these bitmaps and ultimately find the first matching rule. Srinivasan et al. proposed an encoding method called cross-producting that assigns a single number to each disjoint range contained by a classifier field and constructs a look up table for the cross product of the numbers associated with each field [14]. Gupta and McKeown proposed Recursive Flow Classification (RFC) [31], an optimized report of the cross-producting scheme that uses recursive cross-producting tables to reduce the space rations of regular cross producting tables. Furthermore, they map disjoint ranges that are contained by the same set of rules into a single value. RFC's mapping tables use a weaker equivalence relation than Our domain compression technique, so they do not achieve As much compression as we do. Unfortunately, this software

Based re encoding methods are hard to deploy because the Required RAM to perform the re encoding is extremely large. By using TCAMs to achieve re encoding, we conquer this memory issue.

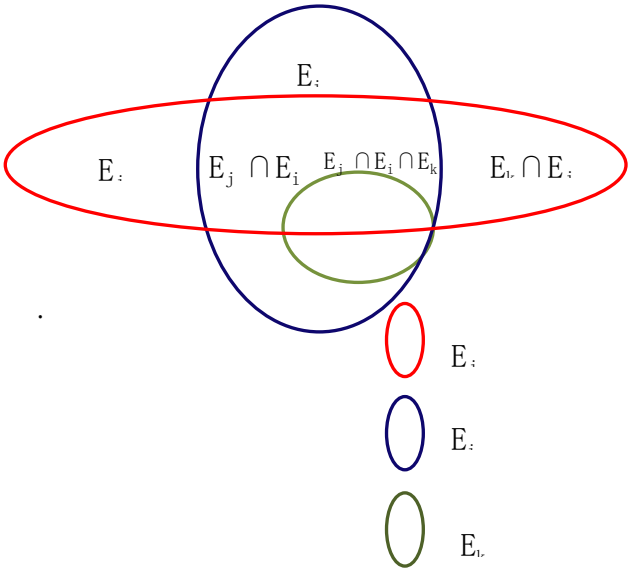
**2.4 TCAM Rule Sets**

First-match TCAM research is to find multi-match results, we require to proof intersections between rules. Studies in [17] show that the number of intersections between real-world rules is drastically smaller than the theoretical upper bound since each field has a limited number of values (e.g., all known port numbers) as a substitute of unconstrained random values. So keeping all the intersection rules is possible. Indices of all those rules that used to create the intersection are stored in a list. SRAM is used to store the list and we call this as a "Match List". We first do a TCAM lookup in a given packet and then use the matching index to recover all matching results with a secondary SRAM lookup as shown in Figure 2. The *extended rule set is formed by* extended rules plus the original rules. The objects in the match list are the indices of rules in the original rule set. As defined in Section 2, the TCAM compatible order entails rules to be ordered so that the first match should record all the matching results in the match list. We first list the relationships between any two different rules $E_i$ and $E_j$, with match list $M_i$ and $M_j$. Exclusive, subset, superset, and intersection, are the four cases with following related requirements:

1. Exclusive $(E_i \cap E_j, = \varphi)$: then $i$ and $j$ can have any order.
2. Subset $(E_i \subseteq E_l)$: then $i<j$ and $M_j \subseteq M_i$ .
3. Superset $(E_j \subset E_l)$: then $j<i$ and $M_i \subseteq M_j$ .
4. Intersection $(E_i \cap E_j \neq \varphi)$: then there is a rule $E_l = (E_i \cap E_j)$ $(l<i, l<j)$, and $(M_i \cup M_j) \subseteq M_l$.

Case 1 is trivial: if $E_i$ and $E_j$ are disjoint, $E_i$ never matches $E_j$ in every packet matching if they may be in any order. For Case 2 where $E_i$ is a subset of $E_j$, $E_i$ will match $E_j$ *in every packet matching* as well, so $E_i$ should be put before $E_j$ and match list $M_i$ should include $M_j$. In this way, packets first matching

$E_i$ will not miss matching $E_j$. Similar operations are required for Case 3. Besides these three cases, partially overlapping rules lead to Case 4. In this case, we need a new rule $E_l$ recording the intersection of these two rules $(E_i \cap E_j)$ placed before both $E_i$ and $E_j$ with both match results included in its match list $(M_i \cup M_j) \subseteq M_l$). Note that the intersection of $E_i$ and $E_j$ may be further divided into smaller regions by other rules (e.g., $Ek$ in Figure 3). In this case, all the smaller regions $(E_i \cap E_j$ and $E_i \cap E_j \cap E_k$ ) have to be presented before both $E_i$ and $E_l$. This can actually be deduced by requirement (4).




.

**Figure 2:Example of intersection of three rules**

Case 1 to 4 covers all the feasible relationships between any two rules. By applying the corresponding operations talked above, we can meet the requirements and get a TCAM compatible order.

---

Extend_rule_set*(R){*
*E*=φ *;*
For all the rule $R_i$ in *R*
*E*=Insert*(*$R_i$*, E);*
return *E;*
*}*
Insert*(x, E){*
for all the rule $E_i$ in *E {*
Switch the relationship between $E_i$ and *x:*
Case exclusive:
continue;
Case subset:
$M_i = M_x \cup M_i$*;*
continue;
Case superset:
$M_x = M_x \cup M_i$*;*
add *x* before $E_i$ *;*
return *E;*
Case intersection:
*If* $(E_i \cap x \notin E$ and $M_x \not\subset M_i)$
add $t = E_i \cap x$ before $E_i$ *;*
$M_t = M_x \cup M_i$
*}*
add *x* at the end of *E* and return *E;*
*}*

**Figure 3: TCAM compatible order code.**

Figure 3 is the pseudo-code for creating a TCAM compatible order. The algorithm gets the original rule set $R=\{R_1, R_2, \ldots, R_n\}$ as the input. Each rule $R_i$ is associated with a match list, which is index of itself *({i}*). The algorithm will output an extended rule set *E* in the TCAM compatible order. The algorithm inserts one rule at a time into the extended rule set E, which is originally empty (the empty set obviously follows the requirements of TCAM compatible order). Next, we will show that after each insertion, *E* still meets the requirements. *Insert(x, E)* is the routine to insert rule *x* into E. It scans every rule $E_i$ in *E* and checks the relationship between $E_i$ and *x*. If they are exclusive, then we can bypass *E*i. If $E_i$ is a subset of x, we just add match list $M_x$ to $M_i$ and proceed to the next rule. If $E_i$ is a superset of x, we should add *x* before $E_i$ *according* to requirement (3) and ignore all the rules after $E_i$ (see the proof in the appendix).

Otherwise, if they intersect, then according to requirement (4), a new rule $E_i \cap x$ needs to be inserted before $E_i$ if it is not presented before. The match list for the new rule is $M_x \cup M_i$. As you can see, we strictly trail the four requirements when adding every new rule, so the generated extended rule set *E* is in the TCAM compatible order. Due to space limitations, we do not go into the details of the deletion algorithm. To better show the algorithm, let's look at the following example in Table 1 which contains three rules. To generate extended rule set *E*, first we insert rule 1. Rule 2 does not intersect with rule 1 so it can be added directly. Now, we have rule 1 followed by rule 2. When inserting rule 3, we find that it intersects with both rule 1 and rule 2, so we add two intersection rules with match list {1, 3} and {2, 3} and put rule 3 at the bottom of the TCAM. The final extended rule set *E* is presented in Table 2.

**Table1 : Example of original rules et with 3 rules**

| S.No | Original rules sets |
|---|---|
| 1 | Tep $SQL_SERVER 1433→$EXTERNAL_NET any |
| 2 | Tep $EXTERNAL_NET 119→ $HOME_NET Any |
| 3 | Tep Any Any → Any 139 |

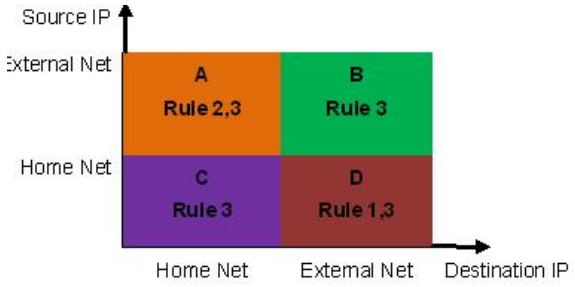



**Figure 5 : Source and destination IP addresses space**

**Table 2 : Extended rule set in the TCAM compatible order**

| Match List | Extended Rules |
|---|---|
| 1.3 | Tep $SQL_SERVER 1433→$EXTERNAL_NET 139 |
| 1 | Tep $SQL_SERVER 1433→$EXTERNAL_NET any |
| 2.3 | Tep $EXTERNAL_NET 119→ $HOME_NET 139 |
| 2 | Tep $EXTERNAL_NET 119→ $HOME_NET Any |
| 3 | Tep Any Any → Any 139 |

### 2.5 Deleting Negation

The scheme there can be used to generate a set of rules in the TCAM compatible order. In this section, describe how to insert them into TCAM. Each cell in the TCAM can take one of three states: 0, 1 or 'do not care'. Hence, each rule needs to be represented by these three states. Generally, a rule includes IP addresses, port information, protocol type, etc. IP addresses in the CIDR form can be represented in the TCAM using the 'do not care' state. On the other hand, the port number may be selected from an arbitrary range. Liu[34] has projected a scheme to competently solve port range problem, so that will not discuss for further.. A more difficult problem for the TCAM is that various IP and port information is in a negation form. As explained in Section 2, each negation consumes many TCAM entries, so in this section, our aim is to take out negation from the rule set to save TCAM space.

Let us first look at the groupings of source and destination IP address spaces as shown in Figure 5 before we present our scheme. Use the rule set in Table 1 as an example, rule 3 applies to all 4 regions since it is "any" source to "any" destination; rule 1 applies to region D because we think $SQL_SERVER is in side $HOME_NET; and rule 2 applies to region A. The regions that contain negation ($EXTERNAL_NET) are region A ($EXTER-NAL_ NET to $HOME_NET), B ($HOME_NET to $EXTER-NAL_NET), and D ($EXTERNAL_NET to $EXTERNAL_NET). Consider region A as an example: the rules in this region are in the form of *"* $EXTERNAL_NET * →$HOME_NET + *"* . Note that * means it could be anything (e.g. "tcp" or "any" or a specific value). $HOME_NET + stands for $HOME_NET and any subset of it such as $SQL_SERVER. If we can extend rules in region A to region A and C, we can swap $EXTERNAL_NET with keyword "any" and now rules are in the format of *"* any * →$HOME_NET + *"*. however, after extending the region, we change the semantics of the rule and this may concern packets in region C. In another word, packet *"* $HOME_NET * → $HOME_NET + *"* will report a match of this rule as well. This problem, however, is solvable because TCAM only informs the first matching result. With this property, we can first take out all the rules applying to region C and put those rules at the top of TCAM. Next, we add a separator rule between region C and region *A: "any $HOME_NET any →$HOME_NET any"* with an empty action list. In this way, all the packets in region C will be matched first and thus ignore all the rules afterwards. With this separator rule, we can now extend all the rules in region A to region A and C. Similarly, rules in region D can be extended to region C and D, rules in region B can be extended to region A, B, C, D. Therefore, we will put all the rules in the following order:

Rules in region C: *"* $HOME_NET + * →$HOME_NET + *"* .

**Table 3.Extended rule set in a TCAM with no negation**

| TCAM Index | TCAM entries | Match list |
|---|---|---|
| 1 | tcp $HOME_NET any →$HOME_NET 139 | 3 |
| 2 | any $HOME_NET any →$HOME_NET any | |
| 3 | tcp $ SQL_SERVER 1433→ any 139 | 3,1 |
| 4 | tcp $ SQL_SERVER 1433→ any any | 1 |
| 5 | tcp any 119 →$HOME_NET 139 | 2,3 |
| 6 | tcp $HOME_NET any →$HOME_NET any | 2 |
| 7 | tcp any any → any 139 | 3 |

Separator rule 1: *"any $HOME_NET any →$HOME_NET any"* .

Rules in region D, specified in the form of region C and D:
*"* $HOME_NET + * →any *"* .
Rules in region A, specified in the form of region A and C: *"* any * →$HOME_NET + *"*.
Separator rule 2: *"any $HOME_NET any →any any"* .
Separator rule 3: *"any any any →$HOME_NET any"*
Rules applying to region B, particular in the form of region A, B, C and D: *"* any * →any *"*
Putting extended rule sets in the above order can be simply attained by first adding all three separator rules to the initiating of the original rule set, then following the algorithm in section 2. If a rule applies to regions A, it will frequently intersect with the separator 1 and generate a rule in region C. If a rule applies region B, then it will intersect with all three separators and create

**Table 4:SNORT rule headers statistics.**

| | Release Date | Rule SetSize | Rules added | Rules deleted |
|---|---|---|---|---|
| 2.0.0 | 4/14/2003 | 240 | - | - |
| 2.0.1 | 7/22/2003 | 255 | 21 | 6 |
| 2.1.0 | 12/18/2003 | 257 | 3 | 1 |
| 2.1.1 | 2/25/2004 | 263 | 6 | 0 |

three intersection rules. After that, we can replace all the $EX-TERNAL_ NET with keyword "any".
Table 3 illustrates the result of mapping the rule set of Table 1into TACM. The first rule in region C is extracted from rule 3 that concerns to all four regions. The second rule is a separator rule. With these two rules, we can swap the $EXTERNAL_NET in rules 3-6 with keyword "any". At the end, there is rule 7 which applies to all the regions. Separator rules 2 and 3 are absent because no rule is in the form of $EXTERNAL_NET to $EXTER-NAL_NET in the original rule set. In this example, by adding only two rules, we can totally remove the $EXTERNAL_NET. Compared to the solution in table 2 which needs up to 4*32 +1 =129 TCAM entries, table 2 which needs up to 4*32 +1 =129 TCAM entries, it is 94.5% of space saving!
The above example is a particular case because there is only one type of negation EXTERNAL_NET) in one field. In a more general case, there can be other than one negation in each field. For example, there could be both !80 and !90 or !subnet1 and !subnet2 in the same field. Our method can be easily extended. If there are *k* unique negations in one field and their non-negation forms do not overlap (e.g., 80 and 90), then we need *k* separators of the non-negation form (80, 90) and they can be in any order. If they intersect, then we need up to *2 k -1* separation rules for this field. For instance, suppose there are !subnet1 and !subnet2, there should be three separation rules applying to subnet1 ∩subnet2, subnet2, and subnet1. *k* is usually a very small number because it is limited by a number of stared subnet.

**Table 5:Statistics of extended ruleset in TCAM compatible order.**

| Version | # of rules in extended set | Single negation | Double Negations | Triple Negations |
|---|---|---|---|---|
| 2.0.0 | 3,693 | 62.334 % | 0.975% | 0 |
| 2.0.1 | 4,009 | 62.484 % | 1.422% | 0.025 % |
| 2.1.0 | 4,015 | 62.540 % | 1.420% | 0.025 % |
| 2.1.1 | 4,330 | 62.332 % | 1.363% | 0.023 % |

In general, if each field *I* needs *ki* separators, then at most of .+1 - 1)) (k ( i separator rules should be added. In our earlier example of removing $EXTER-NAL_NET from source and destination IP addresses, *k1= k1=1*, so we need a total of 2*2-1=3 separator rules.

## III. SIMULATION RESULTS

SNORT [2] rule set is used to test the impression of our algorithm. The SNORT rule set has undergone important changes since 1999. We experienced all the versions after 2.0 that are publicly available. Although each rule set has around 1700-2000 rules, several of the rules share a common rule header. Note that we omitted the versions that share the same rule headers with the previous version. Our mission is to put these rule headers into TCAM as classification rules, and collect the identical

matching rule indices in the match list. Hence, given an incoming packet, with one TCAM lookup and another SRAM lookup, we can execute multi-match packet classification. The second column in Table 5 records the size of extended rule set in the TCAM compatible order. It is nearly 10 times the size of the original rule set, which is well below the theoretical upper bound. This agrees with the findings in [17,32,33].

The number of negations in the extended rule set is important. As shown in Table 5, on average 62.4% of the rules have one negation, 1.295% of the rules have two negations and there are even rules with three negations. In our representation, we assume home network is a class C address that has a 24 bit prefix, so each $EXTERNAL_NET needs 24 TCAM entries. Negation of port, e.g., !80, !21:23 uses 16 TCAM entries. In this setting, a single negation takes up to 24 TCAM entries; a double negation consumes up to 24*24=576 TCAM entries; and a triple negation requires up to 24*24*16=9216 TCAM entries. Therefore, if we directly put all the rules with negation into the TCAM, it takes up to 151,923 TCAM entries as shown in the third column of Table 6. Our negation removing scheme in Section 3 can notably save TCAM space. For the SNORT rule header set, we added 2*3*2*2-1 = 23 separation rules in front of the original rule set because there are four types of negations: $EXTERNAL_NET at source IP, $EXTERNAL_NET at destination IP, !21:23 and !80 at source port, and !80 at destination port. It only adds about 10% extra rules in the extended rule set (4 th column of Table 6). However, with this 10% more rules, we can decrease the number of TCAM entries require d by over 93%. Note that the total number of required TCAM entries is superior than the extended rule set size. This is because some rules have port ranges and consume extra TCAM entries. The range mapping approach in [34] is not used because this approach requires Two extra memory lookups for key translations, and classification speed is our main concern. If a lower speed is acceptable, then we can also include the range mapping technique and the total TCAM entries needed is just the size of extended rule set after removing negations. Each rule is 104 bits (8 bits protocol id, 2 ports with 16 bits each, 2 IP addresses with 32 bits each), which can be rounded up to use a 128 bits entry TCAM. The total TCAM space wanted for SNORT rule header set is 128*8649=135KB. To study the effect of negation, we randomly show a discrepancy the negation percentages in the original rule set. In the SNORT original rule header sets, 89.7% of rules contain single negation and 1.1% of the rules contain double negation. So, we first spotlight on the effect of single negation. Figure 6 shows the TCAM space needed both with and without our negation removing scheme.,

The two schemes perform closely , when the percentage of negation is very low . If we study closely, when the negation percentage is very small (<2%), putting negation directly is better than our scheme while we introduce extra separation rules that may intersect with other rules. However, as the percentage of negation is advanced, the TCAM space needed for “with negation” case grows very fast. In distinguish the curve of our scheme remains flat and thus can save a huge number of TCAM space. For example, when 98% of the convention involve negation, our scheme can save 95.2% of the TCAM space compared to the “with negation” case. This is only for the single negation case. Due to space margins, we do not present result for double negation cases. However, we can make up that the saving would be even higher since each double negation rule requires more TCAM entries.

| **Snort Version** | **With Negation** | | **Negation removed** | | **TCAM Space saved** |
|---|---|---|---|---|---|
| | **Extended Rule set size** | **TCAM entries needed** | **Extended Rule set size** | **TCAM entries needed** | |
| 2.0.0 | 3,693 | 120,409 | 4,101 | 7,853 | 93.4% |
| 2.0.1 | 4,009 | 145,208 | 4,411 | 8,124 | 94.4% |
| 2.1.0 | 4,015 | 145,352 | 4,420 | 8,133 | 94.4% |
| 2.1.1 | 4,330 | 151,923 | 4,797 | 8,649 | 94.3% |

## IV. CONCLUSION:

It is relatively simple to perform packet classification at high speed using large amount of storage or low speed at small amount of memory. In this paper we have classified there are two techniques prefix alignment techniques and multi prefix alignment techniques and consider TCAM re-encoding process has been inducted here. These techniques not only multiple composable and also prefix alignment optimization and create re-encoding scheme. We have implemented our algorithm and conducted extensive experiments and both real-time and multi packet classifiers. The experiment result shows that our techniques achieve 6.74 times more space reductions with transformers.

## REFERENCES

[1] A.Brodnik,S.Carlsson,M.Degermark,S.Pink. “Small Forwarding Tables for Fast Routing Lookups”,*Proc.ACM SIGCOMM 1997*,pp.3-14,Cannes,France.

[2] Abhay K.Parekh and Robert G.Gallager,"A generalized processor sharing approach to flow control in integrated services networkes:The single node case,"IEE/ACM Transactions on networking, vol.1,pp.344-357,June 1993.

[3] Alan Demers, Srinivasan Keshav, and Scott Shenker,"Analysis and simulation of a fair queueing algorithm," Internetworking: Research and Experience, vol. 1, pp. 3-26, January 1990.

[4] Braden et al., "Resource Reservation Protocol (RSVP) –Version 1 Functional Specification," RFC 2205, September1997.

[5] B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," in Proceedings of the Conference on Computer Communications (*IEEE INFOCOMM*), (San Francisco, California), vol. 3, pp. 1248-1256, March/April 1998.

[6]K.Lakshminarayanan, A.Rangarajan and S.Venkatachary,"Algorithms for advanced packet classification with ternery CAMs," in *Proc.ACM SIGCOMM*, 1998, pp. 193 – 204.

[7] M. Waldvogel, G. Varghese, J. Turner, B. Plattner. "Scalable High-Speed IP Routing Lookups," *Proc. ACM SIGCOMM* 1997, pp. 25-36, Cannes, France.

[8] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Transactions on Networking*, vol. 4,3, pp. 375-385, 1996.

[9] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in

Proceedings of the Conference on Computer Communications *(IEEE INFOCOMM*), (San Francisco, California), vol. 3, pp. 1241-1248, March/April 1998.

[10]D.A.Applegate, G.Calinescu, D.S.Johnson,H.Karloff,K.Ligett, and J.Wang, "Compressing rectilinear pictures and minimizing access control lists," in *Proc. ACM-SIAM SODA*, Jan. 2007, pp.1066-1075.

[11] R. Guerin, D. Williams, T. Przygienda, S. Kamat, and A. Orda, "QoS routing mechanisms and OSPF extensions," Internet Draft, Internet Engineering Task Force March 1998. Work in progress.

[12] Richard Edell, Nick McKeown, and Pravin Varaiya, "Billing Users and Pricing for TCP", *IEEE JSAC* Special Issue on Advances in the Fundamentals of Networking, September 1995.

[13] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM* Transactions on Networking, vol. 1, pp. 397-413, August 1993.

[14] E.Spitznagel, D.Taylor, and J.Turner, "Packet classification using Extended TCAMs,"in *Proc.11th IEEE ICNP*,Nov.2003,pp.120-131.

[15]R.Draves, C.King,S.Venkatachary, and B.Zill,"Constructing optimal IP Routing tables," *in Proc.IEEE INFOCOM*, 1999, pp. 88 – 97.

[16] V.Srinivasan and G.Varghese, "Fast IP Lookups using Controlled Prefix Expansion", in Proc. ACM Sigmetrics, June 1998.

[17]M.Kounavis, A.Kumar,HM Vin,R.Yavatkar, and A.Campbell, "Directions in Packet Classification for Network Processors," *NP2 Workshop*, February 2003.

[18] Q.Dong, S.Banarjee, J.Wang, D.Agarwal and A.Shukla,"Packet classifiers in ternary CAMs can be smaller,". In *proc.ACM SIGMANERICS*,2006,pp.311-322.

[19]A.X.Liu and M.G.Gouda, "Diverse firewall design," *in Proc.DSN*,Jun.2004,pp.196-209.

[20]C.R.Meiners, A.X.Liu, and E.Torng, "TCAM Razor:A Systematic approach towards minimizing packet classifiers in TCAMs," in *Proc.15th IEEE ICNP*, Oct.2007,pp.266-275.

[21] ]C.R.Meiners, A.X.Liu, and E.Torng, "Bit weaving:A non-prefix approach to compressing packet classifiers in TCAMs,"Dept.Computer Sci.Eng.,Michigan State Univ.,East Lansing,MI,Tech.Rep.MSU-CSE-09-1,Jan, 2009.

[22] A.Bremler-Barr, and, D.Hendler,"Space-efficient TCAm-based classification using gray coding,"in *Proc.26th Annu. IEEE INFOCOM*,2007,pp.1388-1396.

[23] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES; Dynamic range encoding scheme for TCAM coprocessors,"IEEE Trans.Comput.,vol.57no. 7,pp.902-915, Jul. 2008.

[24] H.Liu,"Efficient mapping of range classifier into ternary-Cam,"in *proc.Hot Interconnects*,2002,pp.95-100.

[25]K.Zheng,H.Che,Z.Wang,B.Liu, and X.Zhang,"DPPC-RE:TCAM- based distributed parallel packet classification with range encoding," *IEEE Tran.Comput*.,vol.55, no.8,pp.947-961,Aug 2006.

[26] A.Bremler-Barr, D. Hay, D.Hendler, and R.Roth, "Layered interval codes for TCAM-based classification," *in proc.IEEE INFOCOM*,2009,pp.1305-1313.

[27]D.Pao, Y.Li, and P.Zhou, "Efficient packet classification using TCAMs,"*Comput.Netw*.,vol.50, no. 18,pp.3523-3535,2006.

[28]D.Pao,P.Zhou,B.Liu, and X.Zhang , "Enhanced prefix inclusion coding filter-encoding algorithm for packet classification with ternary content addressable memory," *Comput.Digital Tech.*, vol.1,no.5,pp.572-580, Sep-2007.

[29]F.Yu,T.V.Lakshman, M.A.MOtoyama, and R.H.Katz,"SSA:A power and memory efficient scheme to multi-match packet classification,"*in proc.ANCS*,Oct. 2005,pp.105-113.

[30]T.V.Lakshman and D.Stiliadis,"High Speed policy-based packet forwarding using efficient multi-dimensional range matching," in proc.ACM SIGCOMM, 1988, pp. 203-214.

[31]P.Gupta and N.Mckeown, "Packet classification on multiple fields",in p*roc.ACM SIGCOMM*, 1999, pp. 147-160.

[32]P.Gupta, N.McKeown "Packet Classification on multiple fields," in *SIGCOMM*, August 2003.

[33]S.Singh, F.Baboesu, G.Vargheese, and J.Wang, "Packet Classification using Multidimensional Cutting,"in SIGCOMM, September 1998

[34]P.Gupta, and N.McKeown, "Algorithms for Packet Classification," *IEEE Network*,March 2001.