Efficient Mapping of Range Classifier into Ternary-CAM

Huan Liu
Department of Electrical Engineering
Stanford University, CA 94305
huanliu@stanford.edu

Abstract

Packet classification is inherently a multi dimensional search problem which is either very computation intensive or memory intensive for software implementation. Thus, hardware based solution is necessary to keep up with gigabit line rate processing. In this paper, we consider using standard Ternary Content Addressable Memory (TCAM) as a hardware classification engine. Traditionally, this approach has been deemed inefficient because ranges have to be broken into prefixes before stored in TCAM, resulting in large expansion. We propose a novel scheme where we can efficiently map ranges into TCAM. Our proposal has no expansion at all, or very little expansion for width constrained application. Our proposal enables high speed deterministic classification using low cost commodity hardware.

1. Introduction

Internet routers need to classify packet based on multiple header fields in order to support functions such as QoS guarantee, traffic engineering and differentiated services.

Packet classification is defined as the action to match the packet with a set of predefined rules, which is called classifier collectively. It involves matching a number of fields in the packet (which we call keys) against the corresponding fields in the rules. We say a packet matches a rule only when all fields match.

The following types of matching are typically required for a layer 2 to layer 4 classifier:

1. Exact match under arbitrary mask: The key is AND with the mask associated with each rule, and then compared with the value for exact match. For example, Ethernet MAC address field and possibly even IP address field [2] fall into this category. In general, the mask bits will be arbitrary, i.e., 1's and 0's are not necessarily contiguous. This kind of matching is particularly hard for software to implement efficiently since a software algorithm has to search each rule linearly. There is no tree structure to exploit to save search time.

2. Longest prefix matching: This is a special case of exact match under mask. Instead of arbitrary mask, this case guarantees that the mask is a contiguous bits of 1's followed by a contiguous bits of 0's. IP address field falls into this category as a result of adoption of CIDR [7].

3. Range matching: A lower limit and an upper limit are specified. If the key falls within the range, it is considered a match. TCP port field and TOS byte field fall into this category. Longest prefix matching is a special case of range matching where the boundary is power of 2. For example, prefix 01xx can be expressed as range 0100-0111.

4. Exact Match: This is a special case of all of the above. The key has to match exactly with the field in the rule.

There are efficient algorithms that solve special cases of the general classification problem. For example, routing lookup only use one field, the destination address, to classify a packet. Many efficient algorithms [6, 10, 1, 9] have been proposed. However, the general classification problem is much harder to solve. Several classification algorithms aiming at software implementation have been proposed [4, 8]. Either their run time or their memory requirement grows quickly with the number of rules supported. It is hard to implement them purely in software and still meet the lookup speed requirement in gigabit routers.

Co-processor has been proposed as an alternative solution. Typically, these processors are based on a general-purpose processor with a number of features implemented in hardware to speedup the lookup operation [3]. The main packet processor sends all rules to the co-processor at system initialization. When new packet arrives, it is passed to co-processor for classification. Effectively, the main processor is offloading the classification function to the co-processor so that it could have further cycle to process new packet. Co-processors typically have low density in terms

Proceedings of the 10TH Symposium on High Performance Interconnects Hot Interconnects (HotI'02)
0-7695-1650-5/02 $17.00 © 2002 IEEE

of number of rules supported, and they are generally very expensive because of limited production.

Another approach for classification is to use standard Ternary Content Addressable Memory (TCAM) hardware search engine. It has several advantages over software based or co-processor based solutions:

1. TCAM's generality enables it to be used in a variety of networking applications. Several vendors such as [5] are offering similar products as a result of huge market demand. Thus TCAM will benefit most from mass adoption. The popularity will encourage further research and development investment to make it faster, cheaper, denser and less power consuming. Coupled with the economy of scale, TCAM will improve faster than any other classification alternatives.

2. Secondly, TCAM architecture is simple and easy to understand. Its standard architecture allows several companies to produce compatible products. Instead of using proprietary classification engine where the architecture is different from one vendor to another, the system designer could standardize on TCAM. Tremendous saving both in system development cost and time to market is possible.

3. TCAM is typically much denser than proprietary architecture because of their regular structure. This translates to smaller package, less number of I/O pins and smaller real estate requirement.

4. TCAM is ideally suited to all matching mentioned earlier except range matching. Especially for matching under mask, TCAM has a clear advantage over all other solutions.

5. TCAM's performance is deterministic. Namely, it takes the same number of cycles to complete every single classification. These are no worse case scenarios that would take longer time.

Despite these advantages, TCAM also has some limitations. One of them is that TCAM could not perform range matching efficiently. A range has to be expanded into prefixes to fit the bit boundary. For example, a range of 1024-65535 needs to be expanded into 6 separate TCAM entries. In general, the number of expansion could be up to $2k$, where k is the width of the field [8]. If more than one range field is specified in a rule, the number of expansion is multiplied. For example, if two range fields are used, each is 16 bits wide, there could be up to $32 \times 32 = 1024$ expansions for a single rule. Clearly, it is not scalable.

In this paper, we present a novel scheme to solve the range matching limitation. Our basic algorithm expands TCAM horizontally (using more bits per entry), and for width limited application, we will also present an algorithm that allows both horizontal and vertical expansion. Since a bit in an entry represents a column in TCAM table, and an entry represents a row, we will use *bit* and *column* interchangeably and use *entry* and *row* interchangeably in the following discussion.

2. Mappping Range to TCAM

Real life classifiers typically exhibit certain unique characteristics. Several relevant observations from [2] are listed here:

1. The transport layer protocol field is restricted to a small set of exact values instead of a generic range.

2. Only 10.2% specifications for transport layer fields are of generic range. Certain ranges are used in a large number of rules. For example, 1023-65535 occurs in 9% of the rules. Assuming the rest 1.2% specifications are all distinct, and assume a classifier has 2000 rules (only 0.7% classifier is larger than 1000), there will only be $2000 \times 1.2\% + 1 = 25$ distinct ranges.

3. It is common for many different rules in the same classifier to share a number of field specifications.

In general, even though the number of rules in a classifier could be large, the number of *distinct* ranges specified for any range field is very limited. Exact match specification also happens frequently for a range field. We suspect the reason is because network operators typically specify rules manually. A human being can only manage a small set to guarantee correct behavior.

These simple observations motivated our proposal. Since the number of distinct ranges is limited, we could use a single bit in an entry to represent each of them. This set of bits is stored in TCAM instead of the original range. For every packet, we have to translate the lookup keys into their bit representation before matching against TCAM. The translation could be completed in a single memory access by direct table lookup.

Our proposed scheme has no row expansion at all. Although the number of bits used could be more than the original field width. This happens when more than k distinct ranges are specified for a k bit wide field. For width constrained application, we will present an algorithm that allows both row and column expansion. The row expansion will be much smaller than that caused by prefix expansion for real life classifier. Although a lookup memory is needed, it is much cheaper and less power consuming than TCAM. We believe we are making the right trade off to allow TCAM to handle large size classifiers.

There are several advantages of our proposed scheme:

1. It requires less TCAM storage space. Because of the limited expansion (or no expansion), our scheme can accommodate a much larger number of rules in a single TCAM table, therefore, reducing system cost and power consumption.

2. It has deterministic execution time. Most of the existing classification algorithms exploits special structure of classification rules. Depending on the particular rule set and the lookup key, the execution time could differ widely. For example, a tree based lookup algorithm's execution time would vary depending on where the lookup key will match in a tree. This is true even for a number of hardware classifier implementations. The worst case execution time could be much larger than the average execution time, making the system performance unpredictable.

Throughout this paper, we will use the simple classifier shown in Table 1 as an example. In the example classifier, 5 rules are specified by the combination of destination IP address and destination port number.

#	Dest IP Addr (IP/mask)	Dest Port Range	Action
1	10.0.0.0/255.0.0.0	$>$ 1023	Deny
2	192.168.0.0/255.255.0.0	50 - 2000	Allow
3	192.169.0.0/255.255.0.0	80 (http)	High Priority
4	172.16.0.0/255.255.0.0	23 (telnet)	Route through port A
5	172.16.0.0/255.255.0.0	21 (ftp)	Rate Limit to 1Mbit/s

Table 1. A simple example classifier

2.1. Range Representation in TCAM

In this section, we first describe how ranges are represented in TCAM. For each range field, we use an n bits vector $B = \{b_1, b_2, \ldots, b_n\}$ to represent it, where n is the number of distinct ranges specified for this field. The B vector for a range R_i has 1 at bit position i, i.e., $b_i = 1$, and all other bits are set to *don't care*. Since B is the actual bit vector stored in TCAM, the bits are tri-valued, as a result, *don't care* is a valid value each bit can take on. An example bit vector representation for the simple classifier in Table 1 is shown in Table 2.

2.2. Lookup key translation

Having just described how ranges are stored in TCAM, we now describe how the lookup key should be translated before matching against TCAM.

A lookup key $v \in [0, 2^k]$ is translated into an n bit vector $V = \{v_1, v_2, \ldots, v_n\}$. Bit v_i is set to 1 if the key v falls into the corresponding range R_i, namely, $v_i = 1$ if and only if $v \in R_i$, otherwise it will be set to 0. For example, in our

Rule #	TCAM rules	
1	10.x.x.x	xxxx1
2	192.168.x.x	xxx1x
3	192.169.x.x	xx1xx
4	172.16.x.x	x1xxx
5	172.16.x.x	1xxxx

Table 2. Rules as stored in TCAM. x denotes *don't care*. The destination address field uses hex number, whereas the port field uses binary number. The right most bit is bit 0, the left most bit is bit 4.

example classifier, a lookup key of 1024 will be translated into 00011 because it falls within both R_1 and R_2. A complete lookup key translation table for the example classifier for each possible lookup key value is shown in Figure 1.

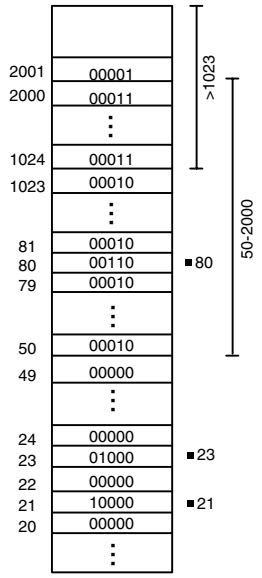

Figure 1. Lookup key translation table for the example classifier

Lookup key translation could be implemented as a direct memory lookup. For a 16 bits wide range field, it requires 64K entries, which could be economically implemented using conventional memory. For typical layer 2 to layer 4 network applications, the range fields such as TCP port, TOS field all have width less than 16 bits. Even for MPLS labels, which are 20 bits, direct memory lookup could be used. If memory size is limited, it could also be implemented using binary search algorithm. The trade off is in lookup time versus memory usage. We omit further discussion on lookup key translation since the algorithms are well known.

Note that the idea of translating ranges into bit repre-

sentation has been used in software classification algorithm [4], but it has not been used to map ranges into TCAM efficiently.

2.3. Exact Match Optimization

Exact match is a special case of range matching. As noted earlier, a large number of the ranges specified in real life classifiers are actually exact matches. In our simple example classifier, rule number 3, 4, 5 are all exact matches.

So far, we have used a single bit to represent each distinct range. In the case of exact match, it is possible to reduce the number of bits used to $log_2(m+1)$ where m is the number of exact matches. The bit representation will contain two parts: B_e for exact matches, and B for all others. $B_e = \{b_1, b_2, \ldots, b_t\}$ is a t bit vector, where $t \geq log_2(m+1)$. For a normal range, its $B_e = 0$ and its B portion is as determined in Section 2.1. For an exact match, its $B = 0$ and its $B_e = i$ if it is the ith exact match. We are effectively assigning code word, and each non-zero value of vector B_e is used to represent a single exact match. Note that $B_e = 0$ could not be used to represent any exact match because it has the special meaning of "none of the exact matches".

In our example, if we use bit 2 and 3 as B_e, the example classifier as stored in TCAM is shown in Table 3. Instead of using 3 bits to represent the 3 distinct exact matches, we used only 2 bits. In general, the saving will be much more significant when the number of distinct exact matches is large, since the number of bits used grows linearly in the standard approach, whereas, it grows logarithmically with exact match optimization.

Rule #	TCAM rule storage	
1	10.x.x.x	xxxx1
2	192.168.x.x	xxx1x
3	192.169.x.x	x01xx
4	172.16.x.x	x10xx
5	172.16.x.x	x11xx

Table 3. Rules as stored in TCAM with exact match optimization

The lookup key translation table needs to be changed accordingly. The lookup key also contains two part: V_e corresponding to all exact mathes, and V corresponding to the rest as described in Section 2.2. $V_e = \{v_1, v_2, \ldots, v_t\}$ is a t bit vector. $V_e = i$ if the lookup key v is equal to the ith exact match, otherwise $V_e = 0$. The resulting bit representation for lookup key value v is a concatenation of V and V_e.

The reason we can use this optimization is because all exact matches are mutually exclusive. Namely, if the lookup key matches one exact match, it will not match another. As long as the ranges involved are mutually exclusive, this optimization could be extended to normal ranges as well.

2.4. A Lookup Example

To illustrate how a packet is classified based on our proposed scheme, we walk through a simple example. Assuming the rules are specified as in Table 1, and it is stored in TCAM as shown in Table 3. A new packet arrives with destination IP address 192.169.10.1 and port number 80. First, the port number is indexed into the lookup key translation table. The resulting $V = 10$ because 80 falls in range 50-2000 but not range >1023. The resulting $V_e = 01$ because 01 is the value assigned to exact match value of 80. V and V_e are concatenated then padded with 0 to form the range lookup key 00110. Together with destination IP address, the lookup will result in a match against rule 3.

3. Controlled Row and Column Expansion

The range mapping algorithm we described in last section will expand the number of column used in TCAM linearly as the number of distinct ranges increases. Since TCAM table typically has a limited number of columns, but has a much larger number of rows, it is desirable that our range mapping algorithm could expand in both row and column direction as the number of distinct ranges grows.

The idea of expanding in both directions is simple. We split the range field into r separate regions; each region is assigned a region code using $log_2(r)$ bits. As a result of the splitting, ranges will be broken into several subranges at region boundaries. Within each region, the range mapping algorithm is used on subranges, namely, each distinct subrange within a region is represented by a single bit. When a range is broken into s sub-ranges, rules with this range need to be represented by s separate entries, each corresponding to one sub-range, resulting in row expansion.

An example is shown in Figure 2. The range field is split into two regions. In region 0, 2 bits are needed to represent the two distinct subranges. Similarly, in region 1, 2 bits are needed to represent all subranges. Note that in both cases, there are 3 separate subranges, but only 2 of them are distinct, therefore only 2 bits are needed. As a result of region splitting, rule 5 is broken up into two separate rules. So, two entries in TCAM are needed to represent rule 5.

Assuming there are n ranges specified for a given range field, there will be at most $2n-1$ regions if we split at every possible end point of the ranges. Therefore, we need at least $log_2(2n-1)$ bits for the range field. This also corresponds to the largest row expansion. At the other end of the solution space, we need at most n bits for the range field, with no

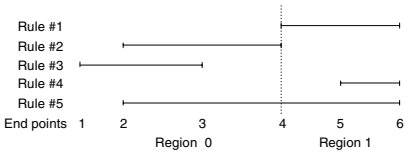

Figure 2. Example showing region splitting

row expansion at all. So, our algorithm is able to provide different solutions based on the number of horizontal bits available in TCAM.

3.1. Heuristic Algorithm for Region Splitting

How to split the range field into regions affects the number of ranges that need to be broken, which in turn affects the degree of row expansion. The goal is to minimize row expansion with a given number of horizontal bits to be used in TCAM.

It is important to note that we only need to consider range end points when deriving optimal way of splitting. For example, in Figure 2, we only need to consider 4 points (excluding the boundary point 1 and 6). Any other points in between will produce less optimal solution.

We used a heuristic algorithm, called Interative Neighbour Merger (INM), that iteratively minimize the number of row expansion while keeping the number of horizontal bits used under a limit. The algorithm keeps a set of points, $rset$, which constitutes the end points of regions. The regions are formed by neighboring end points in the set. We start the iteration with $rset$ containing all end points of all ranges. Effectively, we break at every end point to create at most $2n - 1$ regions. This corresponds to a solution with minimal column expansion and maximum row expansion. The algorithm then tries to pick end point that produces the most number of range cuts, and remove it from $rset$ to cause the two neighboring regions to combine. The two neighboring regions can be combined if the total number of distinct subranges in the combined region is less than the maximum number of horizontal bits available. The algorithm ends when no point in $rset$ could be removed further. It is worth noting that the INM algorithm only considers neighbouring regions. It may be possible to get better result if more combinations can be considered.

In Figure 2 example, assuming we have only a 4 bit wide field. The algorithm starts with 5 regions, which needs 3 bits to encode. In the first two iterations, both end point 3 and 5 will be picked since they both break two ranges each. If we combine the neighboring regions, we will have 3 regions, and at most 2 subranges in each region. Therefore, $\lceil log_2(3) \rceil + 2 = 4$ bits are used, so they can be combined. In the next iteration, either end point 4 or 2 will be picked. Since combining regions around end point 4 will result in more bits than available, only end point 2 can be removed. The algorithm stops in the next iteration since no further end point could be removed.

The pseudo code for the INM algorithm is shown in Figure 3. The symbols used in the algorithm have the following meaning:

- $rset$: The set of points that determines the regions.
- R: The number of regions. It should be $|rset| - 1$.
- $width$: The width of the field in number of bits
- $maxL$: The maximum number of distinct subranges that any region can have
- break(p): The number of ranges that are broken by end point p
- L($rset$): The maximum number of distinct subranges within each region created by end points in rset

```
rset = end points of all ranges
forever
    maxL = width - ⌈log2 R⌉;
    find p such that break(p) = max{break(p'), ∀p' ∈ rset and
                                    L(rset - {p'}) ≤ maxL}
    if ( p = nil ) /* no more point can be removed */
        stop
    else
        rset = rset - {p};
```

Figure 3. Interative Neighbour Merge (INM) algorithm for region splitting

3.2. Expansion Factor Comparison

Because of the width constraint, our INM algorithm will expand rules into more entries just like normal prefix expansion would. We know that prefix expansion is not scalable. Because of potential explosion, only a small number of rules can be stored in TCAM. In this section, we will compare how much expansion is required for each algorithm.

Since no real life classifier is available in public domain, we choose to compare our INM algorithm with regular Prefix Expansion (PE) using randomly generated rules. We first assume that there is only one range field in each rule, and the range field width is 16 bits. Then, we randomly generated 100, 50 and 25 rules each with distinct ranges. Recall that there are only 25 distinct ranges specified in transport layer fields for a real life classifier with 2000 rules as observed in [2], we believe the number of distinct ranges in any range field in any real life classifier will be less than

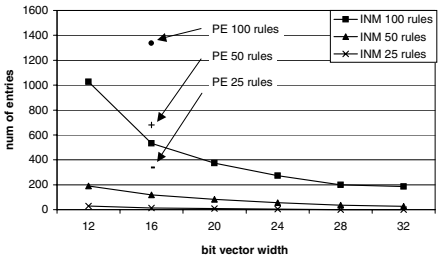

Figure 4. TCAM expansion comparison between Prefix Expansion (PE) and INM algorithm for classifier with single range field. 100, 50 and 25 randomly generated rules are used.

100, therefore, our evaluation gives a good indication of real life performance. The result is shown in Figure 4.

As expected, as the width for the range field is increased, the total amount of row expansion decreases for our INM algorithm. Since our algorithm produces a range of solutions varying in their degree of row and column expansion, the system designer can choose the most suitable one. In terms of degree of row expansion, our INM algorithm clearly outperforms regular range expansion for the real life classifier scenarios we are considering.

The advantage of our algorithm is much more pronounced when we consider rules with two different range fields, for example, rules that specify both the TCP source port and destination port. The number of row expansion for a rule is the same as the product of each individual range field's expansion. To compare the two algorithms, we constructed 100, 50 and 25 rules each with two randomly generated distinct ranges. The result is shown in Figure 5. As expected, our algorithm produces much less expansion.

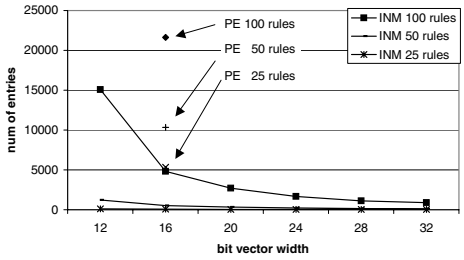

Figure 5. TCAM expansion comparison between Prefix Expansion (PE) and INM algorithm for classifier with two range fields. 100, 50 and 25 rules each with two randomly generated ranges are used.

4. Conclusion

In this paper, we presented a novel scheme to efficiently map ranges inside a rule into TCAM. Traditional approach requires ranges to be expanded into prefixes before stored in TCAM. Since the range could be arbitrary, each expansion could result in $2k$ entries for each single rule, where k is the width of the range field. Our proposed scheme can have no expansion at all when the number of distinct ranges is small, so a large number of rules could be stored in the same TCAM.

Since we use a bit in each entry to represent a distinct range, TCAM width grows linearly as new distinct ranges are added. For width constrained application, we presented an algorithm that allows both horizontal and vertical expansion. It is expected that the number of distinct ranges in a real life classifier is very limited, so our proposed scheme will work very well. Our scheme enables deterministic classification at memory access speed. With our proposal, we believe TCAM could be used to handle classification function better than other alternatives at the most demanding line rate.

References

[1] A. Brodnik, S. Carlsson, M. Degermark, and S. Pink. Small forwarding tables for fast routing lookups. In *Proc. ACM SIGCOMM*, pages 3–14, Cannes, France, 1997.

[2] P. Gupta and N. McKeown. Packet classification on multiple fields. *Proc. Sigcomm, Comp. Commun. Rev.*, 29(4):147–60, Sept. 1999.

[3] S. Iyer, R. Kompella, and A. Shelat. Classipi: An architecture for fast and flexible packet classification. *IEEE Network*, pages 33–41, Mar./Apr. 2001.

[4] T. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proc. ACM Sigcomm*, pages 191–202, Sept. 1998.

[5] Netlogic microsystems. *Ternary Synchronous Content Addressable Memory (IPCAM)*. http://www.netlogicmicro.com/pdf/NL82721.pdf.

[6] S. Nilsson and G. Karlsson. Ip-address lookup using lc-tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–92, 1999.

[7] Y. Rekhter and T. Li. An architecture for ip address allocation with cidr. *RFC 1518*, 1993.

[8] V. Srinivasan, S. Suri, and G. Varghese. Packet classification using tuple space search. In *Proc. ACM Sigcomm*, pages 135–146, Sept. 1999.

[9] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems*, 17(1):1–40, Oct. 1999.

[10] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high-speed ip routing lookups. In *Proc. ACM SIGCOMM*, pages 25–36, Cannes, France, 1997.