A Practical Range Encoding Scheme for TCAMs

Wenjun Li, Xinwei Liu
Peking University & Peng Cheng Lab

Wenxia Le, Hui Li*
PKUSZ & Peng Cheng Lab

Huayu Zhang
2012 Labs, Huawei

ABSTRACT

TCAMs play a central role in the forwarding plane of physical SDN switches. Despite their capability for line-speed queries, they are not well suited for representing rules with range fields. In this poster, we report our latest work on TCAM range encodings, which is not only better in terms of encoding performance, but also easier to understand and more practical to implement. The preliminary evaluation shows that, our proposed encoding scheme has comparable encoding efficiency to the optimal encoding scheme, while achieving an order-of-magnitude improvement on encoding performance over the optimal encoding scheme on average.

CCS CONCEPTS

• Networks → Data path algorithms; **• Hardware → Networking hardware**.

KEYWORDS

OpenFlow, Packet classification, TCAMs, Range encoding

ACM Reference Format:
Wenjun Li, Xinwei Liu, Wenxia Le, Hui Li, and Huayu Zhang. 2019. A Practical Range Encoding Scheme for TCAMs. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342346

1 INTRODUCTION

OpenFlow switches are being deployed to enable a wide spectrum of non-traditional applications in the era of SDN. The OpenFlow switch enforces forwarding policies with multiple *'match-action'* table lookups, which is essentially an extensively studied multi-field packet classification problem [1]. To meet the line-speed requirement of high performance networks, TCAMs have been the dominant implementation of packet classification in physical SDN switches, which enable parallel lookups on all rules for the best match in a single pass.

*Wenjun Li (wenjunli@pku.edu.cn) and Hui Li (lih64@pkusz.edu.cn) are also with the Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School. This work is supported by the PCL Future Regional Network Facilities for Large-scale Experiments and Applications (PCL2018KP001), Shenzhen Municipal Development and Reform Commission (Disciplinary Development Program for Data Science and Intelligent Computing), National Keystone R&D Program of China (2017YFB0803204, 2016YFB0800101), NSFC (61671001) and Shenzhen Research Program (JCYJ20170306092030521).

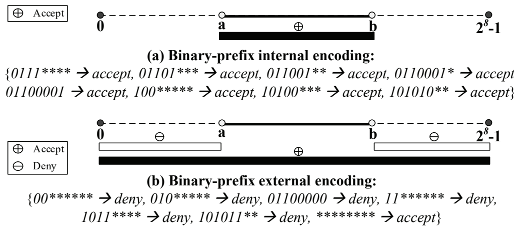

⊕ Accept
0
a
⊕
b
2^8-1
(a) Binary-prefix internal encoding:
{0111**** → accept, 01101*** → accept, 011001** → accept, 0110001* → accept, 01100001 → accept, 100***** → accept, 10100*** → accept, 101010** → accept}
0
⊖
a
b
⊖
2^8-1
⊕ Accept
⊖ Deny
⊕
(b) Binary-prefix external encoding:
{00****** → deny, 010***** → deny, 01100000 → deny, 11****** → deny, 1011**** → deny, 101011** → deny, ******** → accept}

Figure 1: An example of encodings for R_{ab} = [97, 171].

Despite their capability for line-speed queries, TCAMs are not well suited for representing rules with range fields. However, the source and destination port numbers in forwarding rules are usually specified in ranges (i.e., integer intervals), which can not be directly stored in TCAMs. To solve this problem, the common practice is to convert each range into an equivalent set of ternary entries by using range encoding techniques. For example, using internal encoding scheme proposed in [5], an 8-bit width range R_{ab} = [97, 171] can be encoded into 8 prefix entries as shown in Figure 1(a). By exploiting the characteristic of *order of the entries* in TCAMs, external encoding scheme [2] can reduce the encoded entries from 8 to 7 as shown in Figure 1(b). Although the well-known optimal encoding scheme [3] can encode each range with a minimum of ternary entries, its adopted dynamic-programming algorithm [6] is not only complex and hard to understand, but also difficult to implement.

To fill the gap between theory and practice, we report our latest work on range encodings, which achieves similar efficiency in performance of compression as the optimal encoding scheme, but much faster and simpler than the optimal encoding scheme. Instead of adopting complex recursive algorithms as in optimal encodings, all encoded prefixes in our algorithm can be easily obtained in linear time by counting 0/1-bits of the two endpoints. To achieve better compression efficiency, we employ a greedy selection algorithm among the bits of the range endpoints. Additionally, we also give a much simpler proof of the worst-case range expansion that a W-bit width range can be encoded into at most W entries.

Overall, the goal of our work is to design a practical range encoding scheme with suboptimal compression efficiency, but much faster and simpler to implement.

2 DESIGN & PROOF

Before describing the key steps of our algorithm design, we first give some definitions for a W-bit width range R_{ab} = [a, b].

2.1 Definitions

–*Longest Common Prefix (LCP)*: LCP_{ab} is the longest prefix that covers the range R_{ab}. If prefixes are illustrated in a binary trie, LCP_{ab} is the lowest common ancestor of integer a & b.

–*Splitting Endpoint*: For the range R_{ab}, there are two *splitting endpoints* for R_{ab} (one if a = b), where the value of these endpoints are the middle two values of LCP_{ab}.

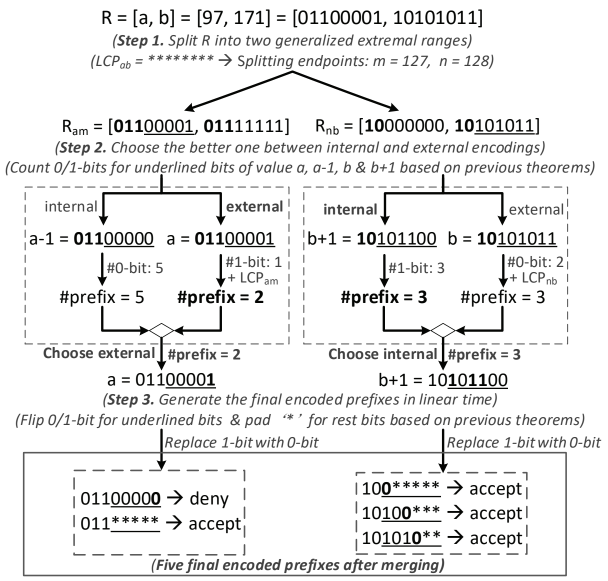

R = [a, b] = [97, 171] = [01100001, 10101011]
(Step 1. Split R into two generalized extremal ranges)
(LCPab = ******** → Splitting endpoints: m = 127, n = 128)
Ram = [01100001, 01111111]
Rnb = [10000000, 10101011]
(Step 2. Choose the better one between internal and external encodings)
(Count 0/1-bits for underlined bits of value a, a-1, b & b+1 based on previous theorems)
internal
external
a-1 = 01100000
a = 01100001
#0-bit: 5
#1-bit: 1 + LCPam
#prefix = 5
#prefix = 2
internal
external
b+1 = 10101100
b = 10101011
#1-bit: 3
#0-bit: 2 + LCPnb
#prefix = 3
#prefix = 3
Choose external
#prefix = 2
a = 01100001
Choose internal
#prefix = 3
b+1 = 10101100
(Step 3. Generate the final encoded prefixes in linear time)
(Flip 0/1-bit for underlined bits & pad '*' for rest bits based on previous theorems)
Replace 1-bit with 0-bit
Replace 1-bit with 0-bit
01100000 → deny
011***** → accept
100***** → accept
10100*** → accept
101010** → accept
(Five final encoded prefixes after merging)

Figure 2: An example using preliminary algorithm.

–*Generalized Extremal Range*: The range R_{ab} is called *extremal range* if a = 0 or b = 2^W-1. More broadly, R_{ab} is called *generalized extremal range* if integer a is the leftmost value of LCP_{ab}, or integer b is the rightmost value of LCP_{ab}.

2.2 Preliminary Design

The key steps of our preliminary algorithm are as follows.

–*Step 1:* If R_{ab} is a *generalized extremal range*, skip to step 2. Otherwise, split R_{ab} into two *generalized extremal ranges*.

–*Step 2:* For each split *generalized extremal range*, choose the better one between internal and external encoding by counting 0/1-bits under the *generalized extremal width* (i.e., excluding the super common bits) of endpoints.

–*Step 3:* Based on the theorems described in [4], the final encoded prefixes can be generated in linear time by recursively flipping each 0/1-bit of the range endpoints - excluding the super common bits - and padding the following bits with '*'.

Thus, take the range R_{ab} shown in Figure 1 as an example, Figure 2 shows the step details of our preliminary encoding scheme, where five prefixes are generated in our algorithm.

2.3 Refined Design

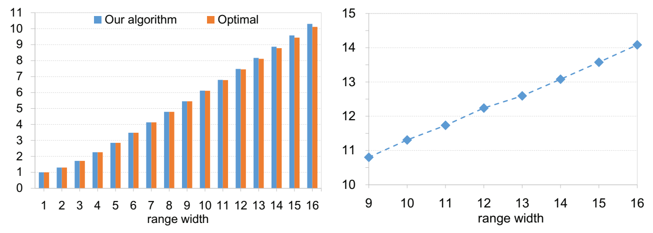
Essentially, the preliminary algorithm simply chooses the better one between the split sub-range and its external complement range, where both of them are encoded with the internal encoding scheme. To further improve encoding efficiency, we can employ the preliminary algorithm to external complement ranges recursively. Due to space and time limitation, we will give more details in our future papers.

2.4 Theorems & Proofs

Finally, we introduce some theorems and proofs on the worst-case range expansion of the W-bit width range R = [a, b]. Although these theorems had been proposed in previous works [2], some much simpler proofs are first described in this poster.

–*Theorem 1:* The range expansion *g(R)* of the *extremal range R* (a=0 or b=2^W-1) satisfies the following upper-bound:

$$g(R) \leq \left\lfloor \frac{W+2}{2} \right\rfloor = \left\lceil \frac{W+1}{2} \right\rceil$$



(a) Average range expansion ratio. (b) Speed-up ratio on encoding time.

Figure 3: Experimental results compared to optimal scheme.

–*Proof 1:* For the binary representation of integer b, it is easy to prove that the number of 1-bits in integer b+1 is at most one more than that in b. Based on the theorems proved in [4], the smallest number of ternary entries that span range [0, b] and range [b+1, 2^W-1] are equal to the number of 1-bits in b+1 and the number of 0-bits in b respectively. Thus, for *extremal range R* = [0, b], the total number of encoded prefixes using internal (i.e., #1-bits in b+1) and external (i.e., #0-bits in b, plus one accept prefix) encoding scheme is at most *W+2* (i.e., *W* bits of integer b, plus at most one more 1-bit, plus one additional accept prefix in external scheme). By choosing the better one, we can obtain the above upper-bound. The proof is similar for *extremal range R* = [a, 2^W-1].

–*Theorem 2:* The range expansion *f(R)* of the general range *R* satisfies the following upper-bound:

$$f(R) \leq W$$

–*Proof 2:* It is not difficult to prove that a W-bit width range R can be split into two *generalized extremal ranges*, where their maximum sub-widths are both *W-1*. Based on the theorem 1, we have: $f(R) \leq \left\lceil \frac{(W-1)+1}{2} \right\rceil + \left\lceil \frac{(W-1)+1}{2} \right\rceil$, so $f(R) \leq W$ or $W+1$ (when W is even or odd). We can further prove that the condition of $f(R) = W+1$ can only happen when both the two split *generalized extremal ranges* are encoded with the external encoding, where the last accept prefix of each subset can be merged (i.e., 0* & 1* → *). Thus, we have $f(R) \leq W$.

3 PRELIMINARY EVALUATIONS

We evaluate the performance of our encoding algorithm with the optimal encoding scheme [3]. Given the range width *W*, we generate all possible ranges with the same probability as follows: [0, 0], [0, 1], ..., [0, 2^W-1], [1, 1], [1, 2], ..., [1, 2^W-1], ..., [2^W-2, 2^W-1], [2^W-1, 2^W-1]. All experiments are run on a machine with AMD Radeon 5-2400G CPU@3.6GHz and 8G DRAM. The operation system is Ubuntu 16.04.

Figure 3(a) and Figure 3(b) show the average range expansion ratio and the speed-up ratio on encoding time over all possible ranges for different width. To reduce the CPU jitter error during evaluation on encoding times, we give results for different width range from 9 to 16 in Figure 3(b). The preliminary experimental results show that our proposed scheme has comparable encoding efficiency to the optimal encoding scheme, while achieving more than 10 times speed-up on encoding performance over the optimal encoding scheme on average. Meanwhile, regardless of the range width *W*, our algorithm can always finish encoding in about 0.1 us, while the optimal encoding takes a few microseconds and grows linearly with the range width *W*.

REFERENCES

[1] Nick McKeown and et al. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM CCR* 38, 2 (2008), 69–74.

[2] Ori Rottenstreich and et al. 2012. Exact worst case TCAM rule expansion. *IEEE Trans. Comput.* 62, 6 (2012), 1127–1140.

[3] Ori Rottenstreich and et al. 2016. Optimal in/out TCAM encodings of ranges. *IEEE/ACM Transactions on Networking* 24, 1 (2016), 555–568.

[4] Baruch Schieber and et al. 2005. Computing the minimum DNF representation of Boolean functions defined by intervals. *Discrete Applied Mathematics* 149, 1-3 (2005), 154–173.

[5] Venkatachary Srinivasan and et al. 1998. Fast and scalable layer four switching. In *ACM SIGCOMM 1998*. 191–202.

[6] Subhash Suri and et al. 2003. Compressing two-dimensional routing tables. *Algorithmica* 35, 4 (2003), 287–300.