Efficient TCAM Encoding Schemes for Packet Classification using Gray Code

Yeim-Kuan Chang and Cheng-Chien Su
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan R.O.C
{ykchang, p7894104}@mail.ncku.edu.tw

***Abstract*—Packet classification is an enabling function in Internet routers for a variety of Internet applications. In order to classify Internet packets into flows, Internet routers must perform searches over a set of filters using multiple fields of the packet as the search key. Because of its speed and simple filter management the Ternary Content Addressable Memory (TCAM) is currently the dominant hardware solution for IP lookups, i.e., a one-dimensional packet classification. To make TCAM the solution for the multi-dimensional packet classification, efficient methods that store the range fields of the classification tables in TCAM are needed. In this paper, we propose a set of novel range encoding schemes based on Gray code. Many range-encoding techniques are used to improve the existing elementary interval-based range encoding schemes. The present experiment's results show that the proposed Gray code-based schemes consume less TCAM storage space than the existing schemes.**

I. INTRODUCTION

Packet classification is needed for a variety of Internet services that require the capability of differentiating various packet flows for suitable processing. Flows are defined by rules that specify some criteria for the field values of the packet headers in incoming packets. In order to classify packets into flows, routers must perform searches over a set of rules using multiple fields of the packet as the search key. Typically, the packet header fields used in filters comprise the following five fields: two prefixes specifying the source and destination IP sub-networks, two arbitrary ranges specifying the source and destination transport-layer specifications, and a singleton value or a wildcard for the protocol number. Routers resolve the flow for a given packet by searching the set of filters for the subset of matching filters against the five header field values of the packets.

Packet classification can be implemented in either software or hardware. Many software approaches were proposed in the literature. However, the software solutions have the disadvantages of un-deterministic run times and memory requirements growing linearly with the sizes of the rule tables. It is a challenge to implement the packet classification purely by software and still be able to meet the search speed requirement in gigabit routers. Readers can refer to [3] for comprehensive surveys on software solutions. In this paper, we focus on hardware architectures, especially, the Ternary Content Addressable Memory (TCAM)-based search engine. The TCAM-based search engine is currently the popular hardware solution because (1) industry vendors are providing cheaper and faster similar TCAM products, (2) TCAM architecture is easy to understand and simple to manage for updating TCAM entries, and (3) TCAM's performance is deterministic (i.e., it takes the same number of cycles to complete a search).

Despite these advantages, TCAMs do suffer from four primary deficiencies: (1) high cost per bit relative to other memory technologies, (2) high power consumption, (3) limited scalability to long input keys, and (4) inefficiency in storing ranges. The cost-to-density-ratio of TCAM has been dramatically improved in recent years. A lower power consumption for TCAMs can be achieved by means of circuit designs that reduce the matchline voltage swing, the switching activity, or the active matchline capacitance [11]. The partitioning techniques proposed in [12][13] can also reduce the TCAM power consumption. However, it can only do so for the prefix fields. A traditional solution for storing ranges in TCAM is the direct range-to-prefix conversion which individually converts each range into multiple prefixes. In the worst case, a W-bit range may require $2(W-1)$ prefixes. A single filter including two port ranges could require $4(W-1)^2$ TCAM entries, or 900 entries for 16-bit port numbers. This is usually referred to as the range-to-prefix blowout. The problem of limited scalability to long input keys can be solved by the range encoding techniques proposed in [7].

Many range encoding schemes were proposed in the literature to solve the range-to-prefix blowout problem [4][5][7]. These range encoding schemes are based on the independent field searches and the two-level SRAM-TCAM architecture shown in Figure 1. The field values in each field of the rule table are independently converted into one or more *field ternary strings* by using a special encoding scheme. The field ternary strings of all field values in a rule are multiplied to obtain one or more *rule ternary strings* that are finally stored in TCAM. Likewise, the address values in the input packet headers must be translated to the intermediate results which are used as the keys to perform the search operations in TCAM. The address-to-intermediate result translation is the additional overhead introduced in the two-level SRAM-TCAM architecture. Thus, the address-to-intermediate result translation table must be done efficiently. In this paper, we do not encode the prefix fields as shown in Figure 1 because prefixes can be stored in TCAM directly. In [4][5], a n-bit vector is used to represent each range field of a rule, where n is the number of distinct ranges specified in this field. In [7], a set of encoding schemes called parallel packet classification

1930-529X/07/$25.00 © 2007 IEEE

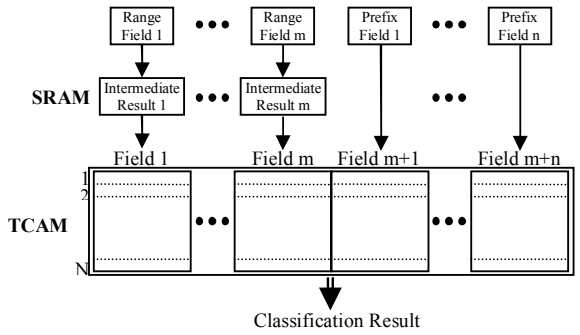


Figure 1.Two-level architecture for packet classification.

encoding (PPCE) is proposed based on the concept of elementary intervals. PPCE improves over the scheme proposed in [5] by exploiting the relationship between ranges. In this paper, we further improve PPCE by using the binary reflected Gray codes [9]. Our experimental results show that the proposed Gray code-based schemes use less TCAM storage than PPCE.

The rest of the paper is organized as follows. Preliminaries and related works are discussed in Section II. The proposed schemes are described in Section III. The results of the performance comparisons are given in Section IV. Finally, the paper is concluded in the last section.

II. Preliminaries And Related Works

We first briefly describe the Buddy code, Gray code, and the elementary intervals needed in the paper. Their formal definitions can be found in [9] and [10]. Then, the existing TCAM encoding schemes that are closely related to the proposed schemes are described.

Buddy Code (BC): The Buddy code follows the traditional number sequence (e.g., 0, 1, 2, 3, …, $2^n - 1$ for n-bit addresses). Based on the Buddy code, the prefix $b_{W-1}\cdots b_h*$ contains the addresses of $b_{W-1}\cdots b_h\overbrace{0\cdots0}^{h}$ to $b_{W-1}\cdots b_h\overbrace{1\cdots1}^{h}$. Each prefix $A = b_{W-1}\cdots b_{h+1}b_h*$ of length $W - h \geq 1$ has a buddy prefix $B = b_{W-1}\cdots b_{h+1}\overline{b_h}*$ such that A and B can be combined into prefix $b_{W-1}\cdots b_{h+1}*$ of length $W - h - 1$.

Binary Reflected Gray Code (BRGC): The binary reflected Gray code is defined recursively as follows. The 1-bit BRGC code is $G_1 = \{0, 1\}$ and the k-bit BRGC code is $G_k = \{0I_0, 0I_1, \cdots, 0I_{2^{k-1}-1}, 1I_{2^{k-1}-1}, \cdots, 1I_1, 1I_0\}$, where $G_{k-1} = \{I_0, I_1, \cdots, I_{2^{k-1}-1}\}$ is the (k–1)-bit BRGC code. For example, the 3-bit BRGC code is G_3 = {000, 001, 011, 010, 110, 111, 101, 100}, where the last inserted bits are underlined.

Elementary intervals: Assume there is a *default range* covering the whole address space. The *elementary intervals*, constructed from the endpoints of a range set G, is $EI = \{E[i] \mid E[i] = [L[i], U[i]]$ for $i = 0$ to $k - 1\}$, where k is the number of elementary intervals in EI. EI must satisfy the following four

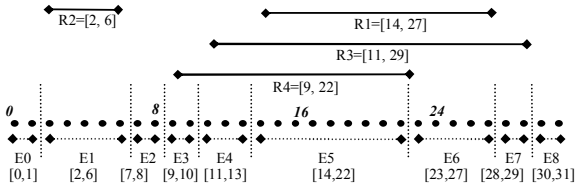


Figure 2.The elementary intervals set EI constructed from $G = \{R1, R2, R3, R4\}$

conditions: (1) $L[0] = 0$ and $U[k-1] = 2^W - 1$, (2) $U[i] = L[i+1] - 1$ for $i = 0$ to $k - 2$, (3) all the addresses in $E[i]$ are covered by the same subset of G, denoted by $G[i]$, and (4) $G[i] \neq G[i+1]$. For example, EI constructed from a set of six 5-bit ranges is shown in Figure 2. There are six *valid elementary intervals* covered by at least one original range in G. The other intervals are *default elementary intervals* covered only by the default range [0, 31].

A. Direct range-to-prefix conversion

In the direct range-to-prefix conversion, each range is individually converted into one or more prefixes. Efficient direct range-to-prefix conversion algorithms can be found in [1] and [10]. For example, the range R = [2, 6] is converted into three prefixes, 001*, 010*, and 0110. In the worst case, the range [1, $2^W - 2$] is split into $2W - 2$ prefixes. For a set of m ranges, the worst-case number of prefixes generated by a direct range-to-prefix conversion algorithm is O(mW) [10].

B. Elementary interval-based encoding scheme

By giving each elementary interval a unique identifier (code), an original range can be encoded by the identifiers of the elementary intervals covered by the range. The ranges encoded by the identifiers of the elementary intervals are called *primitive ranges* [7]. For example, the set of four 5-bit ranges G = {$R1$ = [14, 27], $R2$ = [2, 6], $R3$ = [11, 29], $R4$ = [9, 22]} generates the set of elementary intervals $EI = \{E_0, E_1, \ldots, E_8\}$ shown in Figure 2. First, we assign a unique identifier to each elementary interval. One simple way is to assign the identifiers to the elementary intervals serially (i.e., Buddy code). E_0 gets 0, E_1 gets 1, and so on. Thus, $R1$ can be encoded as a primitive range [5, 6] because $R1$ covers elementary intervals EI_5 and EI_6. Similarly, $R2$, $R3$, and $R4$ are encoded as [1, 1], [4, 7], and [3, 5], respectively. Because there are nine elementary intervals, four bits are needed in the code space. Thus, by using the direct range-to-prefix conversion, $R1$ is converted into two prefixes 0101 and 0110. R2 is converted to 0001, R3 is converted to 01**, and R4 is converted to 0011 and 010*. A total of six 4-bit prefixes are needed. As a comparison, the original direct range-to-prefix conversion generates 17 5-bit prefixes for the same set of ranges. To differentiate this scheme from other elementary interval-based schemes presented in this paper, we call it the *basic elementary interval scheme*.

C. Parallel packet classification encoding (PPCE) scheme

PPCE [7] is also based on elementary intervals but the default elementary intervals are given a common code 0. PPCE divides the original primitive ranges into multiple groups (called *layers*). Depending on the encoding style, the codes assigned to the primitive ranges in one layer may be (1)

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

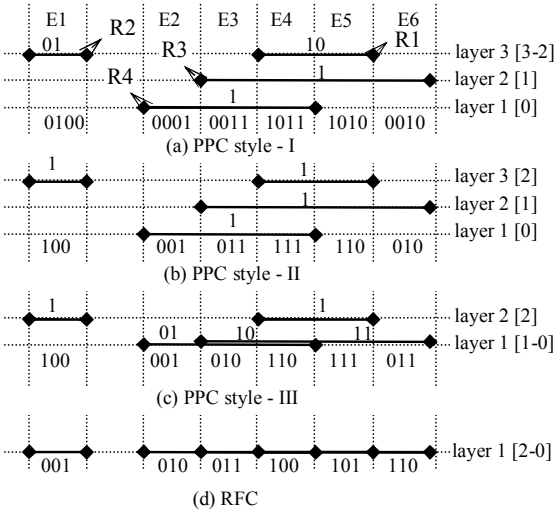


Figure 3. PPCE results for the four ranges in Figure 2.

TABLE I
PPCE MATCH CONDITIONS FOR RANGES IN FIGURE 2.

correspondence	R1	R2	R3	R4
Figure 3 (a)	10**	01**	**1*	***1
Figure 3 (b)	11*	10*	*1*	**1
Figure 3 (c)	11*	10*	*1*	001,*10
Figure 3 (d)	10*	001	011,10*,110	001,*10

independent of, (2) partially dependent on, or (3) completely dependent on the codes assigned to the primitive ranges in other layers.

In the PPCE style-I, all the ranges in one layer must be disjoint. The number of layers required will be equal to the maximum number of ranges that all overlap with each other. The code assignment done in one layer is independent of the other layers. Assume layer i contains L_i ranges. Different non-zero codes are assigned to the L_i address segments covered by the L_i ranges and zero is assigned to addresses that are not covered by any range. $\lceil \log(L_i + 1) \rceil$ bits are needed for layer i consisting of L_i ranges. The intermediate result (identifier) of an elementary interval is obtained by concatenating the codes of the elementary interval in all layers. As shown in Figure 3(a), layer 3 contains R1 and R2; thus, two bits are needed to encode the three address segments created by R1 and R2. The PPCE style-I is the same as the bitmap intersection scheme proposed in [5] if only one range is allowed in a layer.

After we encode the elementary intervals, each primitive range has to be assigned with a *match condition* which is the n-bit ternary string stored in TCAM, where $n = \sum_{i=1}^{m} \lceil \log(L_i + 1) \rceil$ and m is the number of layers. Let R_i be a range in layer i. The match condition of R_i is obtained by setting the corresponding bits to the code assigned to R_i in layer i and the other bits corresponding to the other layers are set to don't care. For example, layers 1, 2, and 3 need 1, 1, and 2 bits, respectively. R1 in Figure 3(a) has the code 10 in layer 3. Therefore, the match condition of R1 is 10**. For an input address p, we can first locate the interval to which address p belongs and obtain the code for that interval to perform a lookup against the match conditions of all ranges. As shown in Figure 3(a), if p is 15, then the corresponding interval is E3. By using the code 0011 assigned to E3, we can find that the matches are R3 and R4.

The PPCE style-II reduces the number of bits required for each layer by inspecting the code dependencies among layers. Two primitive ranges at the same layer can be assigned a common identifier if both ranges are subsets of two disjoint primitive ranges at other layers. For example, in Figure 3(b), the addresses that match R1 must also match R3, but the addresses that match R2 must not match R3. Thus, only one bit is needed in layer 3.

The PPCE style-III reduces the number of bits by reducing the number of layers. It groups the primitive ranges at different layers into one larger layer. The ranges in a layer may be overlapping. However, a range may be represented as more than one match condition. Figure 3(c) shows the final result of the PPCE style-III when R3 and R4 are placed in the same layer. The elementary intervals covered by R1 and R2 in layer 2 are assigned a common identifier "1" based style-II technique. In layer 1, the three disjoint intervals, [E2, E2] covered by R4, [E3, E4] covered by R3 and R4, and [E5, E6] covered by R3, are assigned with the identifiers 01, 10, and 11, respectively. Range R4 needs two match conditions, 001 and *10.

The extreme case of PPCE style-III is to put all the ranges in one single layer, as shown in Figure 3(d), which corresponds to the RFC-like encoding scheme [3]. This extreme case differs from the basic elementary interval-based scheme in that the default elementary intervals are assigned a common identifier 0.

III. PROPOSED RANGE ENCODING SCHEMES

In this section, we improve the existing PPCE encoding schemes [7] by using BRGC instead of the Buddy code. Although the proposed schemes are designed for ranges, they are also applicable to prefixes for reducing the TCAM storage usage, especially in the 128-bit IPv6 source and destination address fields.

A. Elementary interval and BRGC-based range encoding

In this scheme, the default elementary intervals are given a common code and the valid elementary intervals are assigned with the codes based on BRGC. For example, in Figure 2, the codes assigned to the six valid elementary intervals, E1, E3, E4, E5, E6, and E7 are 001, 010, 110, 111, 101, and 100, respectively, in the BRGC sequence. The default elementary intervals, E0, E2, and E8, are assigned with the code 000. As a result, the primitive range R1 is represented as the ternary string 1*1. Similarly, R2 is represented as 001, R3 as 1**, and R4 as, *10 and 111, as shown in Figure 4(a).

B. Enhancing PPCE with BRGC

We modify the PPCE style-II and style-III by using BRGC to assign the identifiers to the valid elementary intervals. We find the ranges that satisfy the following conditions: (1) the ranges contain 2^n valid elementary intervals and (2) the ranges

1930-529X/07/$25.00 © 2007 IEEE

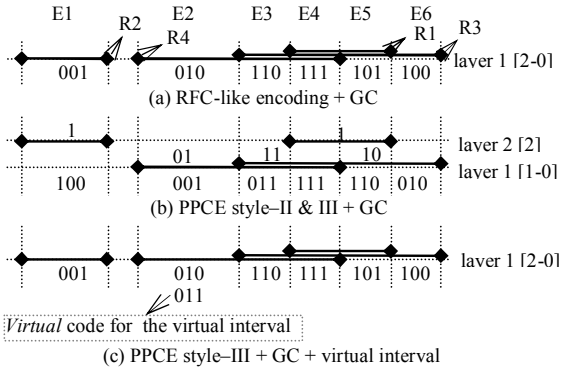


Figure 4.PPCE and GC encoding for ranges in Figure 2.

TABLE II
MATCH CONDITIONS OF THE RANGES IN FIGURE 2 FOR PPCE+GC.

correspondence	R1	R2	R3	R4
Figure 4 (a)	1*1	001	1**	*10, 111
Figure 4 (b)	11*	10* or 100	*1*	**1
Figure 4 (c)	1*1	001	1**	*1*

are overlapped 2^m ($0 \le m \le n$) valid elementary intervals by other ranges. We can assign a sequence of BRGC codes to the 2^n valid elementary intervals.

Figure 4(b) shows one way to assign the BRGC identifiers to the four ranges in Figure 2. Layer 1 needs two bits because ranges R3 and R4 divide the address space into four address segments. The address segments covered by only R4, covered by both R3 and R4, and covered by only R3 are assigned with code 01, 11, and 10, respectively. Layer 1 needs only one bit based on the PPCE style-II. As a result, three bits are needed to represent the match conditions of the ranges as shown in the third row of Table II. The whole address space is mapped onto seven intervals with all the 3-bit codes except 101. For this example, we can reduce the number of match conditions of some range by an optimization as follows. Since no address can be translated to the intermediate value of 101, the search operations will remain correct if the match condition(s) of a range is added with the code 101. As a result, the match condition of R2 can be 100 or 10* if R2 is given an additional code 101. Likewise, the match condition of R4 may be 0*1 and 111 (two prefixes) or **1 (one prefix) if R4 is given an additional code 101. Note that this optimization is different from the *virtual interval* technique described below.

Figure 4(c) shows another way to assign the BRGC identifiers to the four ranges in Figure 2. It is easy to assign the BRGC identifiers such that the match condition of range R1, R2, or R3 can be represented by only one ternary string. However, since R4 covers three consecutive elementary intervals, E2, E3, and E4, then none of the encoding schemes can encode R4 into one ternary string. Because the code 011 is not used, we can view the interval E2 as two intervals by assigning both 011 and 010 to E2, where 011 can be called a *virtual* code. As a result, range R4 can be represented as one ternary string *1* and the ternary strings for other ranges are not changed.

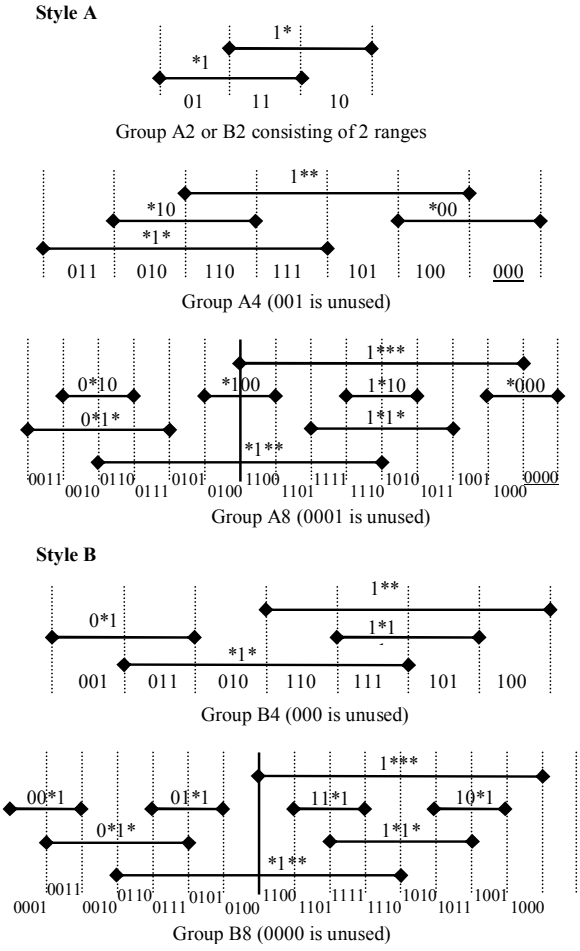


Figure 5. Perfect BRGC range groups.

Based on the identifier assignment shown in Figure 4(c), the four ranges can be viewed as two disjoint range groups, one consists of range R2 and the other consists of three ranges, R1, R3, and R4. Since BRGC identifiers are used, we denote these two range groups as *BRGC range groups*. Our proposed scheme can be formulated as follows.

1. The original ranges are classified into many BRGC range groups. We shall present how ranges can be found to form a BRGC range group later.
2. BRGC range groups are mapped to different layers in a hierarchy such that the BRGC range groups in the same layer must be disjoint and the ranges of the BRGC range groups in the layer are assigned with BRGC codes.

If the BRGC range group is restricted to contain only one range, then the proposed scheme is identical to the PPCE style-I scheme. A BRGC range group is a set of ranges that are encoded by BRGC wherein the match condition of each range in the set is represented by only one ternary string. As shown in Figure 4(c), ranges R1, R3, and R4 are selected to form a BRGC range group, and R2 forms another BGRC range group by itself. Because these two BRGC range groups are disjoint, all the ranges in these two BRGC range groups can be placed

1930-529X/07/$25.00 © 2007 IEEE

in the same layer. Subsequently, we discuss how to find BRGC range groups.

Now, we formally define the *BRGC range group* to be a set of ranges if the following rules are satisfied: (1) no range is disjoint from the other ranges in the group, (2) each range covers exactly 2^c consecutive elementary intervals, and (3) the identifiers assigned to these 2^c intervals can be merged into one ternary string, where c is a non-negative integer. A BRGC range group is called the *perfect* BRGC range group if it covers $2^d - 1$ valid elementary intervals which need $2^d - 1$ distinct identifiers. If a perfect BRGC range group is the only BRGC range group in a layer, then we need d bits for the layer and the unused identifier is assigned to the default elementary intervals. As a result, all 2^d d-bit codes are used and no code is wasted.

We show the perfect BRGC range groups consisting of 2, 4, and 8 ranges in Figure 5. In these perfect BRGC range groups, we assume that the start endpoints or finish endpoints of two ranges are different and the finish endpoint of one range is different from the finish endpoint minus one of another range in the group. The perfect BRGC range groups of 16 or more ranges can be constructed recursively. For example, the two ranges in group A2 cover three valid elementary intervals and are assigned the BRGC identifiers, 01, 11, and 10. The identifier 00 is assigned to the two default elementary intervals. As a result, the two ranges in group A2 can be represented with two 2-bit match conditions, 1* and *1. There are many other possible perfect BRGC range groups when the start endpoints or finish endpoints of two ranges are the same or the finish endpoint of one range is the same as the finish endpoint minus one of another range in the group.

Now, we show the range groups called *imperfect* BRGC range groups that are not perfect BRGC groups. The imperfect BRGC range groups do not utilize all the 2^d d-bit codes if d bits are needed. Therefore, it is better to put imperfect groups in the same layer as other BRGC range groups in order to prevent wasting the unused codes. For example, Figure 6 shows two imperfect BRGC range groups, *C* and *D*. Group *D* needs four bits to encode the nine elementary intervals it covers. We can see that six 4-bit codes are wasted aside from the one code used for the default intervals. Group *C* is also an imperfect BRGC range group. If group *C* and one single range are put into the same layer as group *D*, no extra bit is thus needed.

Subsequently, we introduce two code assignment techniques to facilitate the search for range groups so that each range can be represented with one ternary string.

C. Many-to-one code assignment

The *many-to-one code assignment scheme* achieves the goal of using only one ternary string to encode a range by assigning more than one code to an elementary interval. We have shown an example that assigns two codes to an elementary interval in Figure 4(c). The many-to-one code assignment scheme is applied when the second rule of the BRGC range group defined above is not satisfied, that is, not all the ranges in the group cover exactly 2^c consecutive elementary intervals. It is not possible to merge the codes assigned to the elementary intervals covered by a range into one ternary string if the number of the intervals is not 2^c. If

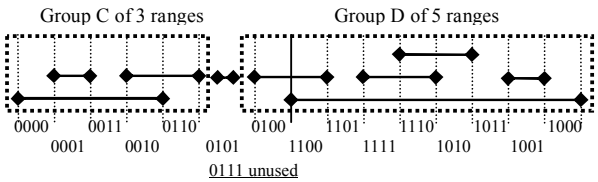


Figure 6.Imperfect BRGC range groups

there are so many ranges that do not cover exactly 2^c consecutive elementary intervals, determining which elementary intervals are given the unused codes may be difficult. Therefore, we cautiously employ this scheme as follows. For a group of ranges, we try to assign more unused codes to the existing intervals covered by these k ranges only if there are less than k ranges that cover $2^c - s$ consecutive elementary intervals. The values of k and s can be varied. In this paper, we set k and s to 1, as in the example of Figure 4(c).

D. One-to-many code assignment

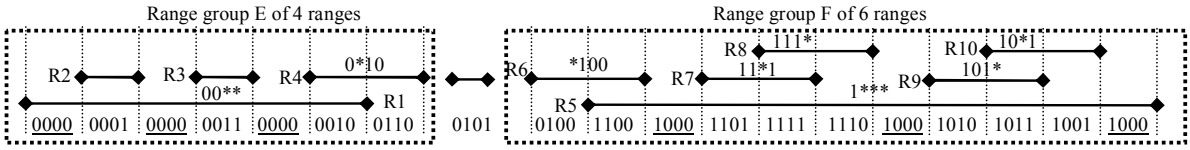
It is a common case that one range may overlap with other ranges. Consider a range set *S* in which the range *R* completely encloses the other ranges in *S*. Based on the proposed encoding schemes described above, *S* cannot form a perfect BRGC range group because one code, usually 0, is reserved for the default interval. Thus, range *R* must not cover the power of 2 consecutive elementary intervals. However, *S* may form an imperfect BRGC range group. Therefore, we develop the *one-to-many code assignment scheme* which can be used to find an imperfect BRGC range group for this common case. The one-to-many code assignment scheme extends the use of a common code from the default elementary intervals to the valid elementary intervals by assigning a common code to the valid elementary intervals covered by the same subset of original ranges. Consider the example shown in Figure 7. In range group E, R1 encloses R2 and R3 and then intersects R4. The three elementary intervals that are covered only by R1 are given the same code 0000, as shown underlined. There are three other elementary intervals that are covered by R1. These three elementary intervals are assigned with codes, 0001, 0011, and 0010. Based on BRGC, the elementary interval only covered by R4 is given 0110 so that the match condition of R4 can be represented as a ternary string 0*10. The code assignment for the range group F can also be done by the one-to-many code assignment scheme. As a result, only four bits are needed for the 11 ranges illustrated in Figure 7.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed BRGC-based TCAM encoding schemes and compare them with the existing algorithms. Under the same classification tables, we evaluate storage requirements in terms of SRAM and TCAM.

We use ClassBench [8] to generate the rule tables and extract their source and destination ports as the range sets to form a 2-D rule tables. Since the range sets for the source and destination ports we generated are large, they lose the characteristics of the real rule tables. In other words, all the ranges are heavily intersected with each other. They can be treated as randomly generated range sets. Likewise, because

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.



The unused code 0111 can be used for default interval.

Figure 7.The imperfect BRGC range groups by the enclosure encoding technique.

TABLE III

TCAM STORAGE REQUIREMENTS OF 100, 500, AND 1000 2-D RULES IN TERMS OF *TOTAL*(*NUM*/*LEN*), WHERE NUM IS THE NUMBER OF TCAM ENTRIES AND LEN IS THE NUMBER OF BITS IN EACH TCAM ENTRY, AND TOTAL = NUM×LEN.

Rule Table	FW100	FW500	FW1000
Direct	618,464 (19,327/32)	3,218,656 (100,583/32)	6,315,520 (197,360/32)
EIEBC	53,792 (3,362/16)	673,800 (33,690/20)	1,783,804 (81,082/22)
EIEGC	39,216 (2,451/16)	520,800 (26,040/20)	1,432,200 (65,100/22)
Bitmap	20,000 (100/200)	500,000 (500/1,000)	2,000,000 (1,000/2,000)
PPCE	16,500 (100/165)	404,000 (500/808)	1,533,000 (1,000/1,533)
PPCGC	15,500 (100/155)	370,500 (500/741)	1,422,000 (1,000/1,422)

TABLE IV

SRAM STORAGE REQUIREMENTS FOR 2-D RULE TABLES IN KBIT.

Rule Table	FW100	FW500	FW1000
Direct	0	0	0
EIEBC	1,024	1,280	1,408
EIEGC	1,024	1,280	1,408
Bitmap	12,800	64,000	128,000
PPCE	10,560	51,712	98,112
PPCGC	9,920	47,424	91,008

the results for ACL, IPC, and Firewall are similar, we only show the results for Firewall.

The evaluated schemes are the direct range-to-prefix conversion (Direct), the elementary interval-based scheme using Buddy code (EIEBC) and the basic elementary interval-based scheme using BRGC (EIEGC), the bitmap intersection scheme (Bitmap), PPCE style-II or III which performs the best (PPCE), and the proposed modified PPCE scheme using BRGC and virtual intervals (PPCGC). Table III shows the TCAM storage requirements in terms of number of TCAM entries and the size of a single TCAM entry.

As expected, Direct consumes the most TCAM storage space among all the tested methods. Bitmap, PPCE, and PPCGC need the least number of TCAM entries, but they need longer TCAM entries. The size of TCAM entries for EIEBC and EIEGC is only 22 bits. EIEGC and PPCGC need the least number of TCAM bits. The difference of the required TCAM space between EIEGC and PPCGC decreases when the size of the rule table increases.

Now, we calculate the SRAM space required for translating input addresses to the code values (i.e., the intermediate results). Since the port numbers are currently 16-bit values, we implement the address-to-intermediate result translation by using an array of 64k entries for each of the source and destination port fields. The entry size of the translation array needs to be as wide as the TCAM entries corresponding to the tested scheme. Table IV shows that Bitmap, PPCE, and PPCGC require much more SRAM than the other schemes. The use of SRAM for Bitmap, PPCE, or PPCGC grows linearly as the size of rule table increases. The size of SRAM needed for EIEBC and EIEGC does not increase much as the size of the rule table increases.

V. CONLUSIONS

In this paper, we improve the existing range encoding schemes for TCAMs by using the binary reflected Gray code. To evaluate our scheme, we use ClassBench to generate classification tables and present the performance results. Our results show that the proposed Gray code-based schemes perform better than the traditional Buddy code-based schemes in terms of TCAM and SRAM memory usages.

REFERENCES

[1] M.D. Berg, M.V. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer Verlag, 1997.

[2] H. J. Chao, "Next Generation Routers," *Proceeding of the IEEE*, vol. 90, no. 9, pages 1518-1558, 2002.

[3] P. Gupta and N. McKeown, "Packet classification on multiple fields,"Comput. Commun. Rev., vol. 29, pp. 147–160, Oct. 1999.

[4] P. Gupta, N. McKeown, "Algorithms for packet classification," IEEE Network (2001) 24–32.

[5] T. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," Comput. Commun. Rev., vol. 28, no. 4, pp. 203–214, Oct. 1998.

[6] H. Liu, "Efficient mapping of range classifier into ternary-CAM," in IEEE Symp. High Performance Interconnects (Hotl'02), 2002.

[7] J. V. Lunteren and T. Engbersen, "Fast and Scalable Packet Classification," IEEE Journal on Selected Areas in Communications, Vol. 21, No. 4, May 2003.

[8] D. E. Taylor and J. S. Turner, "ClassBench: A Packet Classification Benchmark," in Proc. INFOCOM 05, March 2005.

[9] E. M. Reingold, J. Nievergelt. and N. Deo, *Combinatorial Algorithms*, Englewuod Cliffs, NJ: Prentice-Hall, 1977.

[10] Y.K. Chang, "A 2-Level TCAM Architecture for Ranges", IEEE Trans. on Comp., Vol. 55, no. 12, pp. 1614 – 1629, Dec. 2006.

[11] N. Mohan and, M. Sachdev, "Low Power Dual Matchline Ternary Content Addressable Memory," Proc. IEEE ISCAS 2004, Vancouver, pp. 633-636, May 23-26, 2004.

[12] F. Zane, G. Narlikar, and A. Basu, "CoolCAMs: Power-Efficient TCAMs for Forwarding Engines," in Proc. INFOCOM 03, March 2003.

[13] Y.K. Chang, "Power-Efficient TCAM Partitioning for IP Lookups with Incremental Updates", Lecture Notes in Computer Science, Vol. 3391, pp. 531 – 540, Jan. 2005.

1930-529X/07/$25.00 © 2007 IEEE