Space-Efficient TCAM-based Classification Using Gray Coding

Anat Bremler-Barr
School of Computer Science, the Interdisciplinary Center
Email: bremler@idc.ac.il

Danny Hendler
Department of Computer Science, Ben-Gurion University
Email: hendlerd@cs.bgu.ac.il

***Abstract*— Ternary content-addressable memories (TCAMs) are increasingly used for high-speed packet classification. TCAMs compare packet headers against all rules in a classification database in parallel and thus provide high throughput unparalleled by software-based solutions.**

TCAMs are not well-suited, however, for representing rules that contain range fields. Such rules have to be represented by multiple TCAM entries. The resulting *range expansion* can dramatically reduce TCAM utilization.

The majority of real-life database ranges are short. We present a novel algorithm called *short range gray encoding* (SRGE) for the efficient representation of short range rules. SRGE encodes range borders as *binary reflected gray codes* and then represents the resulting range by a minimal set of ternary strings. SRGE is *database independent* and does not use TCAM extra bits.

For the small number of ranges whose expansion is not significantly reduced by SRGE, we use *dependent encoding* that exploits the extra bits available on today's TCAMs. Our comparative analysis establishes that this hybrid scheme utilizes TCAM more efficiently than previously published solutions.

The SRGE algorithm has worst-case expansion ratio of $2W-4$, where W is the range-field length . We prove that any TCAM encoding scheme has worst-case expansion ratio W or more.

I. Introduction

Packet classification is an indispensable building block of numerous Internet applications in the areas of routing, monitoring, security, and multimedia [1], [2], [3]. Network routers employ packet classification schemes to streams of incoming or outgoing packets in order to determine how each packet should be handled. The routers use a *classification database* that consists of a set of *rule*s (a.k.a. *filter*s). Each such rule specifies which actions to apply to each packet that *matches* the rule, actions such as whether the packet should be forwarded or dropped, whether it should be logged or not, etc.

In addition to specifying which actions to take, each rule also specifies a pattern that determines which packets will match it. These patterns are specified based on packet header fields, such as the source/destination addresses and the source/destination port numbers. A *key* is constructed from the corresponding fields of each packet header and is compared against the database rules. If the key matches the rule, then that rule is used to determine how the packet should be handled. Packet classification is often a performance bottleneck in the network infrastructure. It is therefore important to design packet classification solutions that scale to millions of key search operations per second.

Ternary content-addressable memory (TCAM) devices are increasingly used in the Industry for performing high-speed packet classification. A TCAM is an associative memory hardware device which can be viewed as an array of fixed-width entries. Each TCAM entry consists of ternary digits, each of which can assume the values 0, 1, or $*$ ('don't-care'). When storing a classification database, each TCAM entry is associated with a single classification rule and specifies a pattern of packets that match the rule. Typically this leaves a few dozens of *extra bits* in each entry which can be used by range-encoding schemes (see Sections III-B, III-C).

TCAMs enable parallel matching of keys, such as those derived from packet headers, against all entries. They can thus provide high throughput that is unparalleled by software-based solutions. When a key matches multiple TCAM entries, the TCAM returns the index of the first matching entry. This index is then used to locate the information specifying which actions to apply to the packet.

A single ternary digit in a TCAM device requires 10-12 transistors compared to only 4-6 required by a single SRAM bit [4]. This explains why TCAM devices are much more expensive than SRAMs and are a key contributor to the cost of gigabit line cards. As the number of TCAM devices deployed worldwide is growing quickly, improving TCAM memory utilization has important practical implications.

A significant obstacle to the efficient use of TCAMs for packet classification is the fact that they are not well suited for representing rules that contain *range fields*, such as port fields. In general, such rules must be represented by multiple TCAM entries. The resulting *range expansion* can dramatically reduce the utilization of TCAM memory.

Traditionally, a range rule is converted to an equivalent set of prefix rules (each of which can be directly stored to a single TCAM entry) by using a *prefix expansion* technique originally proposed by Srinivasan et al. [5]. Prefix expansion can be highly inefficient, however, and was shown to cause a 6-fold and more expansion of ranges that appear in real-life databases [1].

A. Our Approach and Contributions

This paper presents a novel algorithm for the efficient representation of range rules in TCAM-based classification databases. Our work is motivated by the following observation, resulting from the analysis of a large real-life classification

database consisting of more than $223K$ rules: *the large majority of the ranges used by classification rules are relatively short*. Specifically, the length of 60% of the unique ranges and 40% of all ranges is less than 20.

We present a novel algorithm called *short range gray encoding* (SRGE) for the efficient representation of short range rules. SRGE works by encoding numbers (both range-borders and search keys) using *binary-reflected Gray code* (BRGC) and then covering each resulting range by a set of ternary strings.

A *Gray code* is a binary encoding of a contiguous range of integers such that the codes of any two adjacent numbers differ by a single bit. An n-bit BRGC is constructed recursively by reflecting an $(n-1)$-bit BRGC (see Section IV for more details). It is exactly this reflection property that is being used by the SRGE algorithm as it allows to efficiently cover ranges by using ternary strings that contain 'don't-care' symbols.

To illustrate the benefits of using BRGC codewords as opposed to a regular binary representation, consider a range $R = [i, i+1]$ of length 2. As the BRGC codes of i and $i+1$ differ by a single bit, R can be represented by a single TCAM entry that contains a single don't-care digit (in the single position where i and $i+1$ differ). In contrast, an average of 50% of length-2 ranges over numbers encoded by the regular binary representation require *two* TCAM entries.

Similarly to BRGC, previously published algorithms [6], [7], [8] also manage to reduce the expansion of rules (as compared to prefix expansion) by representing ranges as a set of arbitrary ternary values rather than as a set of prefixes. However, all these algorithms require extra bits, a small number of each is available in each TCAM entry, for representing *all ranges*. To the best of our knowledge, SRGE is the first algorithm that significantly reduces range expansion without resorting to the use of extra bits.

Another novel idea used by SRGE to farther reduce expansion is the representation of ranges by a set of possibly *overlapping* ternary strings. All previously published algorithms use a set of non-overlapping strings to represent a range. We emphasize that, although some of the entries representing a range may overlap, *the SRGE algorithm only requires a single TCAM lookup*. If a key falls inside a range R, then the lookup will return the first matching entry that belongs to the cover of R.

As our empirical results show (see Section VI-C), SRGE achieves a reduction of 25% in the number of redundant TCAM entries required to represent ranges (when compared with prefix expansion) for a big majority of the range rules. A similar reduction in expansion is obtained for randomly-generated short ranges.

To further improve TCAM utilization, we use the extra bits that are typically available in TCAM entries by employing a database-dependent *hybrid-SRGE* scheme. The high-level idea is to automatically find the range rules whose SRGE encoding requires the highest number of redundant TCAM entries and to assign each of these a single extra bit.

Our tests show that the *hybrid-SRGE* scheme succeeds in reducing database expansion caused by range rules from a factor of 2.3 (achieved by using prefix expansion) to a factor of only 1.03, better than any other previously published algorithm. We emphasize that, unlike previously published algorithms, Hybrid-SRGE significantly reduces range expansion for a large majority of the ranges without having to use extra bits. We consequently believe that our algorithm will scale well into the future as the number of ranges used by classification databases continues to increase.

On the more theoretical side, we observe that the *worst-case range expansion* of SRGE is $2W-4$, where W is the size of range fields. This slightly improves over prefix expansion, which was shown to have worst-case expansion of $2W-2$ (see, e.g., [8]). Moreover, we prove that if no extra bits are used then the worst-case expansion of *any* range encoding scheme is at least W.

II. Terminology and Problem Statement

In this section we introduce the terminology we use throughout the paper and define the problem addressed by it. We follow the notation of [8] wherever appropriate.

A packet header consists of fields, each of which is a bit string. A *key* is a collection of K fields from the packet header. Keys are matched against classification *rules* stored in *entries* of a *classification database*.

Rules consist of K fields matching the corresponding key fields. Packet P matches rule R if each of P's K key fields matches the corresponding field of R. Each rule field f can specify one of three types of *matches*.

1) *Exact match*: field f matches key field g if they are equal.
2) *prefix match*: a *prefix* is a string of bits. Field f is a prefix match for key field g if g is a prefix of f.
3) *range match*: a *range* is a contiguous interval of integers $[s, e]$, where s and e are W-bit numbers and $s \leq e$. Key fields matched by ranges are port fields of constant size W (typically 16 bits). The *length* of a range is the number of integers it contains.

This paper deals with classification databases that reside in a TCAM device. A TCAM entry consists of ternary digits, each of which can assume the values 0, 1 or 'don't care', denoted by $*$. Each TCAM entry is wide enough to contain the concatenation of all the key fields, possibly having room for some *extra bits*.

If a rule consists solely of fields that specify exact and/or prefix matches then it can be represented by a TCAM entry in a straightforward manner: a field representing an exact match is stored in the TCAM entry as is; a field representing a prefix match is padded with the appropriate number of don't-cares in the least significant digits.

In general, rules containing one or more range fields cannot be represented by a single TCAM entry and *range encoding schemes* are used to encode each range as a set of TCAM entries. An encoding scheme maps each range R to a set of TCAM entries that represent it, called the *cover set of R*.

The *expansion* of a range is the number of TCAM entries in its cover set. The *range expansion factor* of an encoding scheme E, denoted R_E, is the maximum size of a cover set when using E, the maximum taken over all possible ranges. The expansion factor is a function of W.

A widely used scheme for range encoding converts a range to a set of prefixes, each of which is stored at a separate TCAM entry. For example, for $W = 3$, the range $[1, 6]$ can be represented by the cover set $\{001, 01*, 10*, 110\}$ with range expansion 4. It is known that the range expansion factor of this scheme over W-bit ranges is $2W - 2$ [1].

Let D be a classification database. We let $n(D)$ denote the total number of rules in D. We let $n_E(D)$ denote the number of TCAM entries required to represent D using scheme E. Clearly $n_E(D) = n(D)$ if D contains no range rules. We let $F_E(D)$ denote D's *database expansion factor* using E, defined as $\frac{n_E(D)}{n(D)}$. In other words, $F_E(D)$ is the relative increase in the number of entries required to represent D in TCAM using scheme E.

We let $r(D)$ denote the number of range rules in D. We let $nr_E(D)$ denote the number of TCAM entries required to represent all of D's range rules using encoding scheme E. The *range redundancy* of an encoding scheme E for a range R is the number of additional, redundant, TCAM entries required to represent R when represented by E. We let $FR_E(D)$ denote D's *range redundancy factor* using E, defined as $\frac{nr_E(D)-r(D)}{r(D)}$. In other words, $FR_E(D)$ is the average number of redundant TCAM entries required to encode range rules of D using scheme E.

In this paper we propose an encoding scheme that reduces both the database expansion factor and the range redundancy factor for real-world classification databases.

III. Related Work

The issue of using TCAM devices for packet classification has received considerable attention from the research community over the last few years. A key question dealt with by researchers in this regard is that of improving the utilization of TCAM memory. This issue was considered both from the algorithmic [6], [7], [8], [9] and the architectural [10] perspectives.

Spitznagel, Taylor, and Turner, introduced *Extended TCAM* (E-TCAM) [10], which implements range matching directly in hardware in addition to reducing power consumption by over 90% relative to standard TCAM. While this may represent a promising long-term solution, it seems that changing the ternary nature of TCAM entries while maintaining reasonable per-bit cost and addressing scalability issues will not be accomplished in the near future.

In this section we briefly describe prior algorithmic work that is related to the issue of TCAM range representation.

A. Prefix Expansion

The traditional technique for range representation, originated by Srinivasan et al. [5]), is to represent a range by a set of prefixes, each of which can be stored by a single TCAM entry. The worst-case expansion ratio when using prefix expansion for W-bit fields is $2W - 2$. The problematic range is $R_w = [1, 2^w - 2]$. It is easily seen that the smallest set of prefixes that covers R_w is the following: $\{01*^{w-2}, 001*^{w-3}, 0001*^{w-4}, \cdots, 0^{w-1}1, 10*^{w-2}, 110*^{w-3}, \cdots, 1^{w-1}0\}$. [1]

As observed by [1], a single rule that includes two 16-bit range fields could, in the worst-case, require $(2 \cdot 16 - 2)^2 = 900$ entries.

B. Database-dependent Range Encoding

Database-dependent encoding of ranges [6] makes use of extra bits, available in TCAM entries, for encoding ranges that occur in the database more efficiently. In such schemes, the encoding of a range may depend on the number of occurrences of that range (and of other ranges) in the database.

The number of unique range fields in today's classification databases is around 300. As observed by [8], this number is anticipated to continue to grow in the future. The number of extra bits per TCAM entry may vary according to the configuration of the device, but is typically a few dozen bits. It is therefore clear that the aforementioned simple scheme is not scalable.

Liu [6] proposed hierarchical encoding schemes to alleviate this problem. The encoding scheme of [6], however, may result in high expansion. Lunteren and Engbersen [7] suggested to use hierarchical encoding for compressing general TCAM rules. They present several versions of their scheme. However, the version that may reduce the expansion of range rules considerably complicates the task of incrementally updating the database.

In addition to making updates more expensive, hierarchical dependant encoding requires extra logic so that the appropriate search key fields can be matched against all possible ranges. To maintain high throughput, either special-purpose hardware must be used or a pre-computed table must be stored in memory, whose size increases quickly with the number of ranges. These problems restrict the scalability of hierarchical dependant encoding.

In contrast, our hybrid-SRGE scheme encodes the vast majority of ranges without using extra bits and uses dependent encoding only for a very small number of ranges. It thus avoids the above problems.

C. Database Independent Range Encoding

Independent encoding techniques are techniques that encode each range independently of the distribution of ranges in the database. Lakshminarayana et al. present a clever algorithm for the independent encoding of ranges, based on the concept of *fence encoding* [8]. Their technique, called Database Independent Pre-Encoding (DIRPE), represents ranges by sets of arbitrary ternary strings and is based on the use of extra bits. Unlike the algorithms described in Section III-B , DIRPE

[1] While this is the smallest set of prefixes required to cover R_w, we observe that this is *not* the smallest set of *ternary strings* that can cover R_w. In fact R_w can be covered by the following set of W ternary strings: $\{01*^{w-2}, *01*^{w-3}, **01*^{w-4}, \cdots, 1*^{w-2}0\}$.

is an independent encoding scheme and extra bits are never assigned to any particular range.

The efficiency of DIRPE is a function of the number of extra bits that are available. When the number of available extra bits decreases, the DIRPE-encoding of *all* ranges becomes less efficient. In contrast, the SRGE algorithm is an efficient independent encoding scheme that does not require extra bits.

Hybrid-SRGE encodes the few ranges whose expansion is not improved by SRGE by assigning an extra bit to each of them. A similar hybrid approach is employed by DIRPE for decreasing its expansion. However, unlike in our scheme, in the case of DIRPE the assignment of extra bits to frequently-occurring ranges increases the expansion ratio of all other ranges.

We evaluated our hybrid-SRGE scheme on the same database that was used to evaluate DIRPE in [8]. Our results establish that hybrid-SRGE outperforms hybrid-DIRPE on that database by achieving a range expansion factor of 1.03, compared with a factor of 1.12 achieved by DIRPE. This superior expansion factor is achieved by using less than 40% of the extra bits used by hybrid-DIRPE.

D. Minimizing Boolean Expression

We observe that the problem of minimizing TCAM range expansion is, in fact, a special case of the problem of minimizing the size of Disjunctive Normal Form (DNF) expressions. This connection between the two problems was unnoticed prior to this work.

Mapping the TCAM range expansion problem to the problem of minimizing DNF expressions is done as follows. The variables in the DNF expression correspond to the W bits representing a range. The boolean DNF expression is the representation of the range. A range is expressed as a sum of *minterms*, each of which represents a number [2] and the goal is to find the minimal sum-of-products of the expression. For example, let us consider the range $R = [10, 11]$. Let b_0 denote the units digit and let b_1 denote the tenths digits. The sum of minterms corresponding to R is b_1b_0+b_1b_0' and the minimal sum of products is b_1 which corresponds to the ternary string $1*$.

DNF minimization is a well studied problem. The Karnaugh maps technique can be used to solve instances of the problem involving up to 5 variables and the Quine-McCluskey algorithm can provide a general solution. As the problem is NP-complete, the Quine-McCluskey algorithm is impractical when the DNF formula involves a large number of variables.

A recent paper by Schieber et al. presents a linear time algorithm for finding the minimum size DNF expression corresponding to any range of binary-coded numbers [11]. They also show that the worst-case expansion of ranges over binary-coded numbers to arbitrary ternary strings is $2W - 4$, slightly better than the $2W - 2$ expansion factor achievable when only prefixes may be used.

[2]A minterm is also called a *standard product* or *canonical product term*. This is a term in which each variable appears exactly once.

The rest of the paper is organized as follows. Sections IV and V describes the SRGE algorithm and hybrid scheme, respectively. In section VI, our empirical evaluation of SRGE establishes that it reduces the expansion of a large majority of the ranges as compared to prefix encoding. In Section VII, we prove that correctness and performance claims for the SRGE algorithm. In section VIII we prove that any encoding scheme has worst-case expansion ratio at least W. Conclusions and open questions are discussed in section IX.

IV. SRGE: Efficient Encoding of Short Ranges

In this section we describe the *Short Range Gray Encoding* (SRGE) algorithm for the efficient representation of short range rules in TCAM devices.

A *Gray code* is a binary encoding of a contiguous range of integers such that the codes of any two adjacent numbers differ by a single bit. SRGE uses a specific Gray code called the *binary-reflected Gray code* (BRGC). An n-bit BRGC is generated recursively as follows. The first 2^{n-1} code words are constructed by prefixing 0 to all the $(n-1)$-bit BRGC code words; the last 2^{n-1} code words are constructed by prefixing 1 to the reflected (i.e. listed in reverse order) list of $(n-1)$-bit BRGC code words. It is exactly this reflection property of the BRGC code that allows our algorithm to minimize the size of range cover sets.

The pseudo-code of the SGRE algorithm is shown in Figure 1. The SRGE-cover procedure receives a range $[s_b, e_b]$ of binary numbers, of size 2 or more. It returns a set of ternary strings covering the SRGE codes of all the numbers in this range. In the following description of the pseudo-code of the SRGE-cover procedure, we refer the reader to the illustrations of Figure 2. We let $\mathcal{T}$ denote the full binary tree of height W shown in these illustrations.

First, the BRGC codes of s_b and e_b are computed (and stored into variables s and b, respectively); the least-common-ancestor (LCA) of s and b in $\mathcal{T}$, denoted p, is also computed (step (1), see Figure 1, (1)). Computing BRGC codes (not shown) is very simple and can be implemented efficiently in software as follows. The most significant digit is unchanged; any other digit i of the BRGC code is constructed by taking the exclusive-or of binary digits i and $i + 1$.

The range $[s, e]$ is split by p into two sub-ranges: one in p's left subtree and the other in its right subtree. The right and left borders of these sub-ranges are computed in step (2) and stored to variables pl and pr, respectively (see Figure 1, (2)).

Without loss of generality, assume that the sub-range in p's left subtree, $[s, pl]$, is no longer than the one in p's right subtree. A cover set of prefixes, denoted *prefixes*$_1$, that covers the BRGC codes of all the numbers in $[s, pl]$, is computed in sub-step (3.1). Now the reflection property of BRGC coding is used for minimizing the size of the cover set as follows (see Figure 1, (3)). The digit in each of the prefixes in *prefixes*$_1$ corresponding to p's right/left edges is changed to $*$ (sub-steps (3.2), (3.3)). The reflection property of BRGC coding guarantees that *prefixes*$_1$ now covers a mirror sub-range of $[s, pl]$ with regard to p.

SRGE-cover($[s_b, e_b]$) **returns** a set of ternary strings covering the range $[s_b, e_b]$

(1) $s \leftarrow$ BRGC encoding of s_b, $e \leftarrow$ BRGC encoding of e_b, $p \leftarrow$ least common ancestor of s and e
(2) $pl \leftarrow$ rightmost leaf in p's left subtree, $pr \leftarrow$ leftmost leaf in p's right subtree.
(3) **if** $||[s, pl]|| \leq ||[pr, e]||$ # The other case is symmetric
  (3.1) *prefixes*$_1 \leftarrow$ prefix cover of $[s, pl]$
  (3.2) $i \leftarrow$ digit position corresponding to p's left/right edges
  (3.3) $\forall q \in$ *prefixes*$_1$: set q's i'th digit to $*$. # *prefixes*$_1$ now also covers the mirror of $[s, pl]$ with regard to p
  (3.4) $s' \leftarrow pr + ||[s, pl]||$ # $[s', e]$ is the sub-range of $[pr, e]$ not covered by *prefixes*$_1$
  (3.5) **if** $||[s', e]|| = 0$ **return** *pefixes*$_1$
We still need to cover $[s', e]$
(4) $p' \leftarrow$ least common ancestor of s' and e, $pl' \leftarrow$ rightmost leaf in p''s left subtree, $pr' \leftarrow$ leftmost leaf in p''s right subtree
(5) **if** $||[pr', e]|| \geq ||[s', pl']||$: # Case I
  (5.1) *prefixes*$_2 \leftarrow$ prefix cover of $[pr', e]$
  (5.2) $i \leftarrow$ digit position corresponding to p''s left/right edges
  (5.3) $\forall q \in$ *prefixes*$_2$: set q's i'th digit to $*$. # *prefixes*$_2$ now also covers the mirror of $[pr', e]$ with regard to p'
  (5.4) return *prefixes*$_1 \bigcup$ *prefixes*$_2$
(6) **else** # CASE II: $|[s', pl']| > |[pr', e]|$
  (6.1) *prefixes*$_2 \leftarrow$ prefix cover of $[pr', e]$
  (6.2) $q \leftarrow$ the prefix corresponding to p' left subtree
  (6.3) **return** *prefixes*$_1 \bigcup$ *prefixes*$_2 \bigcup \{q\}$

SRGE-construct-key(b) **returns** a search key for value b
return BRGC encoding of b

Fig. 1. Pseudo-Code for the SRGE algorithm

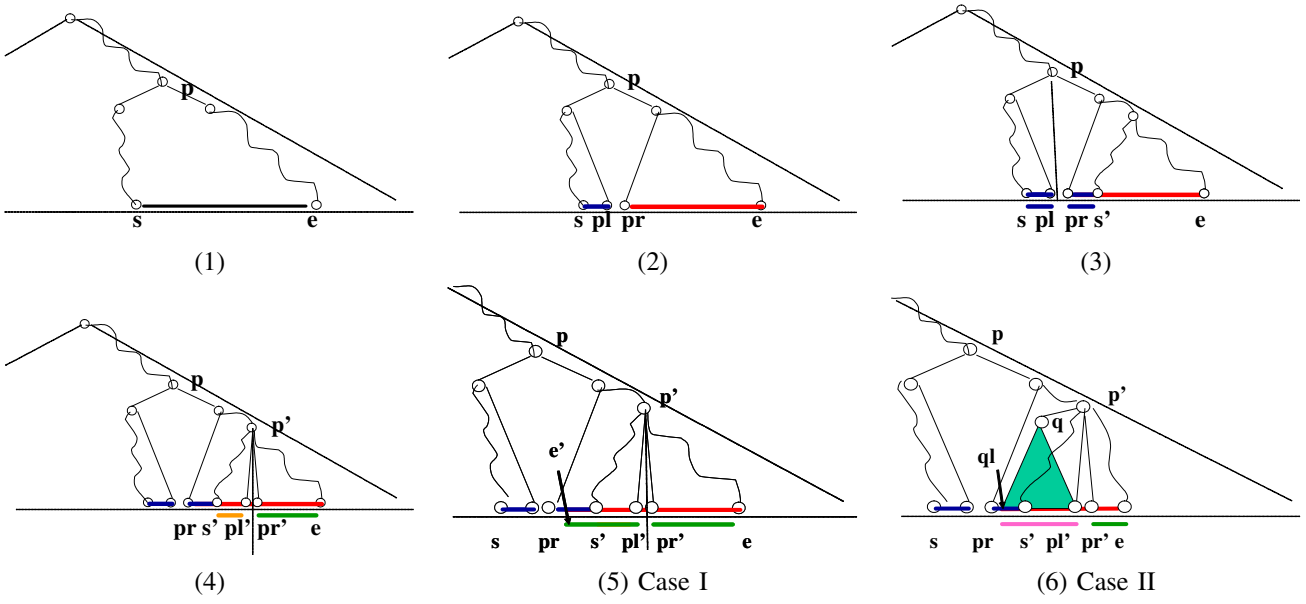

Fig. 2. The steps of the SRGE algorithm

If the input range $[s, e]$ is completely covered by *prefixes*$_1$, then this cover set is returned (sub-step (3.5)). Otherwise, the residue of the right sub-range still needs to be covered. Let the sub-range $[s', e]$ be that residue.

Similarly to step (3), the LCA of s' and e in $\mathcal{T}$, denoted p', is computed (step (4)). The range $[s', e]$ is partitioned by p' into two sub-ranges: one in its left subtree and the other in its right subtree. The left and right borders of these sub-ranges are computed and stored to variables pl' and pr', respectively (see Figure 1, (4)). Now there are two cases to consider. I

1) The sub-range $[pr', e]$, in the right subtree of p', is no shorter than the sub-range $[s', pl']$ in the left subtree of p'. In this case we can cover $[s', pl']$ by reflecting the prefix cover set of $[pr', e]$ around p'. Note that we are guaranteed that all covered numbers are within the input range. This is accomplished by step (5), see Figure 1,

(5).

2) Otherwise, reflecting the cover set of $[pr', e]$ is not enough to cover $[s', pl']$. In this case $[s', pl']$ is covered by the prefix corresponding to the left child of p'. Once again, we are guaranteed that all the numbers covered by this prefix are within the input range. This is accomplished by step (6), see Figure 1, (6).

We note that, in general, the ternary strings in the cover set produced by the SRGE-cover procedure may overlap each other. We emphasize that, although some of the entries representing a range may overlap, the SRGE algorithm only requires a single TCAM lookup. If a key falls inside a range R, then the lookup will return the first matching entry that belongs to the cover of R.

The SRGE-construct-key procedure receives a (binary) header field b and returns the corresponding key with which to search for rules matching the field (see Figure 1). The key is simply the SRGE encoding of b.

Figure 3 demonstrates how the SRGE algorithm covers the range $[6, 14]$.

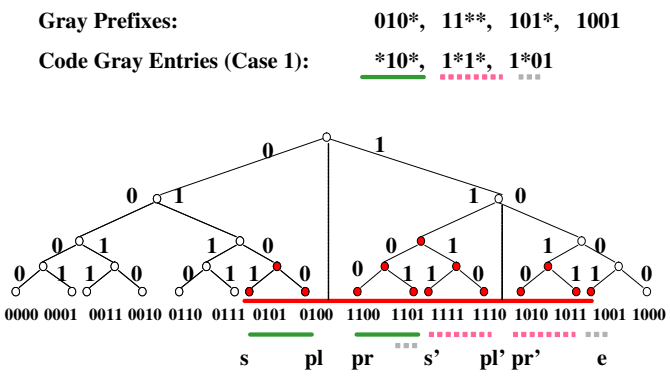

Fig. 3. SRGE example: covering the range $[6 - 14]$.

V. Hybrid-SRGE

The size of TCAM entries is typically larger than the size of the classification rules stored in them. This leaves a number of extra bits which can be used by range-encoding schemes. To further improve TCAM utilization, we use these extra bits by employing a database-dependent *hybrid-SRGE* scheme similar to that described in [6].

The high-level idea is to assign a single extra bit to each of the ranges whose TCAM entries consumption is highest under SRGE encoding. More specifically, let x denote the number of extra bits available in every TCAM entry. Hybrid-SRGE works as follows. First, it computes a list of the unique ranges that occur in the database, sorted in decreasing order of the overall number of redundant entries they require under SRGE encoding. By 'overall number' we mean the total number of redundant entries that are required by the SRGE representation of all the occurrences of the range in the database. Then, each of the first x ranges in this list is dealt with by using a standard

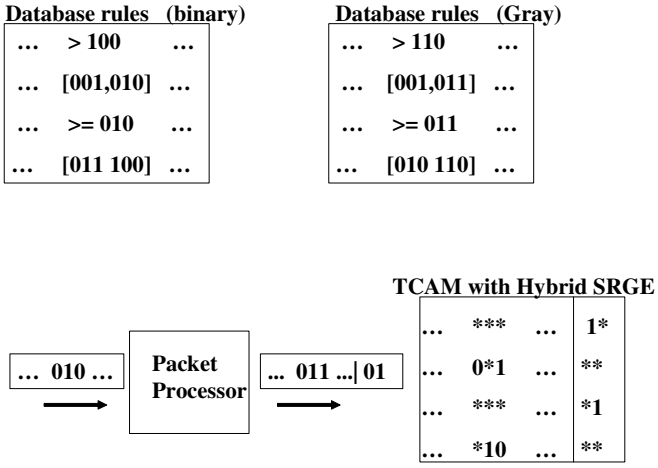

Fig. 4. Example of the hybrid-SRGE scheme. Binary ranges > 100, > 010 are mapped using dependent coding, while the other ranges are encoded using BRGC. This example only displays the values of a single packet field. The binary-encoded field value '010' is transformed to BRGC code '011'. The first extra bit in the corresponding search key is set, as value '011' is included in range ≥ 011 and the first extra bit is allocated to this range.

database-dependent encoding that assigns a single extra bit to it. We call these ranges the x *heaviest ranges*.

To exemplify hybrid-SRGE, consider the well-known range $\leq$ 1024 that encapsulates all registered ports. This is the heaviest range under SRGE. Hybrid-SRGE assigns the first extra bit (bit 1) to the heaviest range. Thus bit 1 is assigned to the range ≤ 1024. The extra bit is used as follows. Extra bit 1 of each entry that contains a source port field with the range ≤ 1024 is set to 1. Extra bit 1 of all other entries is set to $*$. [3]

As for search keys, bit 1 of a search key is set to 1 if the key falls within range R and to 0 otherwise. This guarantees that a key whose source port field is outside R will never match R and that a key whose source port field is inside R may match R, depending on the values of its other fields. Assigning an extra bit to a range results in expansion ratio 1 (i.e. no expansion) for that range.

See Figure 4 for an example illustrating the architecture of hybrid-SRGE.

VI. Experimental Results and Comparative Analysis

We evaluated the efficiency of the hybrid-SRGE scheme on both real-life and random databases.

Our real-life database is a collection of 126 separate files originating from various applications (e.g., firewalls, acl-routers, intrusion prevention systems) collected over the year 2004. This database is the union of the two databases that were used by [1] and [8]. We are grateful to the authors of these papers for sharing these databases with us. The database comprises a total of 223K rules which contain 280 unique

[3]The range $\leq$ 1024 can appear also in the destination port field. All the occurrences of this range in the destination port field are represented independently by an additional extra bit, extra bit 2.

ranges. Prefix expansion resulted in an expansion factor of 2.6. Overall, $\sim$ 26% of the database rules contained range fields. Excluding the single range ≥ 1024, $\sim 14\%$ of the rules contained ranges.

A. Short Ranges

Figure 5 displays the distribution of range lengths in our database. More than 60% of the unique ranges that appear in the database have length less than 20 and 22% percentage of the total number of unique ranges have length 2.

However, only 22% of the *total* number of ranges have length less than 20. The huge difference between the fractions of short unique ranges and short ranges is largely caused by a single rule that appears very frequently in a single database file. To mitigate the effect of such anomalous files, Figure 5 displays also the fraction of short ranges when only ranges that appear in at least *two files* are taken into account. Measured this way, the fraction of short ranges grows to $\sim 40\%$.

Why do short ranges occur so frequently in real-life classification databases? This phenomenon primarily results from the fact that ranges are commonly used for matching the source-port and destination-port fields. Port numbers are allocated by IANA [12] and it is often the case that ports that belong the the same protocol family are assigned consecutive numbers. For example, the port number of $snmp$ is 161 and the port number of $snmptrap$ is 162. Many classification rules need to match the $snmp$ protocol family and, consequently, use the range $161-162$. Typically, each application is assigned a small number of ports which can thus be matched by a short range.

There is, however, also a small number of applications that use a wide range of ports. As two examples, Microsoft's DirectX gaming uses ports in the range $2300-2400$ and Real Audio uses ports in the range $6970-7170$ [4].

A second type of long ranges are ranges that partition all ports to two general categories [13]. Key examples are the well known rules ≤ 1024 and > 1024 that partition ports to the sets of registered ports and dynamic/private ports, respectively.

B. Evaluating SRGE on Random Databases

We compare the efficiency of different range representation algorithms by comparing their *database expansion factor* and *range redundancy factor*. The database expansion factor of database *D* using scheme *E* is the relative increase in the number of entries required to represent D in TCAM using scheme E. The *range redundancy factor* of database *D* using scheme *E* is the average number of redundant TCAM entries required to encode range rules of D using E. See Section II for the precise definitions of these metrics.

Clearly, a perfect encoding scheme will achieve a database expansion factor of 1 (i.e. no expansion). The range encoding redundancy factor focuses only on range rules and quantifies the number of extra TCAM entries that is required to represent them. A perfect encoding scheme will clearly have a range encoding redundancy factor of 0.

[4]For confidentiality reasons, the above examples are not derived from our database.

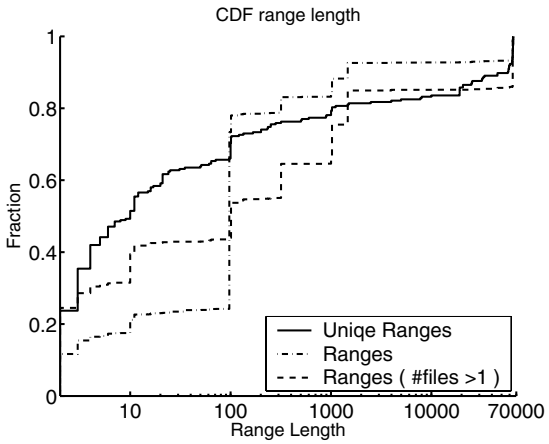

Fig. 5. Distribution of range lengths, calculated as a fraction of 1) the total number of unique ranges, 2) the total number of ranges 3), the total number of ranges that appeared in at least 2 files

Figure 6 compares the efficiency of SRGE and prefix encoding on a database that consist of ranges whose length is chosen randomly and uniformly from the interval $[1, 2^x]$. It shows the reduction in range expansion obtained by SRGE as compared with prefix encoding, as a function of x. SRGE is significantly more efficient for short ranges and reduces the range expansion of databases in which all ranges are shorter than 32 by $\sim 16\%$. For random databases in which the range length is chosen randomly and uniformly from the interval $[1, 2^{16}]$, the average expansion reduction is $\sim 5.5\%$.

C. Evaluating hybrid-SRGE on Real-Life Databases

Table I summarizes our evaluation results on real-life databases. The hybrid-SRGE scheme reduces the database expansion factor to 1.03 by using only 14 extra bits.

We tested hybrid-SRGE also on the database that was used for testing DIRPE in [8]. Using 14 extra bits, we obtained an expansion factor of 1.03. This is a significant improvement as compared to the hybrid-DIRPE algorithm which, as reported in [8], achieved an expansion factor of 1.12 on the same database by using 36 extra bits. Figure 7 displays the contribution of the first x "heaviest ranges" (excluding the range ≥ 1024) to the total expansion.

For each range, we calculate its contribution as the product of the number of times it appears in the database and the number of entries required to represent it by prefix expansion. As can be seen, less than 10 of the heaviest ranges contribute $\sim 92\%$ of the expansion. Taking into account only ranges that appear in more than one file does not change the result significantly. The expansion factor of 1.03 was obtained by hybrid-SRGE by coding the 12 heaviest ranges with extra bits. [5] Our calculations show that SRGE reduces the redundancy caused by all the ranges that are not assigned extra bits by $\sim 25\%$.

[5]Two of the 10 unique ranges appear in both the source and destination port fields. We thus needed a total of 12 extra bits to encode these 10 ranges. Taking into account the additional 2 bits that were required for encoding the ranges ≥ 1024, we get a total of 14 extra bits.

	Algorithm	Extra bits	Redundancy	Expansion
Avr. DB	Binary	2	3.37	1.82
Avr. Files	Binary	2	3.85	1.34
Avr. DB	hybrid-DIRPE[8]	36	-	1.12
Avr. DB	hybrid-SRGE	14	1.2	1.03

TABLE I

EXPANSION AND REDUNDANCY FACTORS USING PREFIX ENCODING (BINARY), HYBRID-DIRPE AND HYBRID-SRGE USING EXTRA BITS. THE BINARY ENCODING FACTORS ARE COMPUTED ASSUMING THAT THE RANGE ≥ 1024 FOR BOTH THE SOURCE AND DESTINATION PORT FIELDS IS ASSIGNED AN EXTRA BIT.

VII. SRGE CORRECTNESS AND PROPERTIES

In order to prove the correctness of the SRGE algorithm, we need to show that the algorithm covers correctly any range. Let $R = [s, e]$ be a range sent as input to the SRGE-cover procedure. Let p, s', pl, pr, p', pl', pr' be as in Figure 2, e' be the reflection of e w.r.t. p', q be the root of p's left subtree, and ql be the leftmost leaf in q's subtree. Note that pl' is the rightmost leaf in q's subtree. In step 3, $[s, pl]$ and its reflection $[pr, s']$ are covered (the other case is symmetric). If $|[s', e']| > 0$, then the covering set is extended in case I (step 5, by covering $[pr', e]$ and its reflection $[e', pl']$) or case II (step 6, by covering $[ql, pl']$ and $[pr', e]$).

Hence, we need to prove that the three following cases of covering $[s, e]$ hold:

1) $|[s, pl]| = |[pr, e]|$: $[s, e] = [s, pl] \cup [pr, s']$ (where s'=e)
2) Case I: $[s, e] = [s, pl] \cup [pr, s'] \cup [e', pl'] \cup [pr', e]$
3) Case II:$[s, e] = [s, pl] \cup [pr, s'] \cup [ql, pl'] \cup [pr', e]$

The first case is straightforward. In cases I and II, since $[s, e] = [s, pl] \cup [pr, s'] \cup [s', pl'] \cup [pr', e]$, we only need to prove the following two claims concerning the range $[s', pl']$.

Claim 1: In case I, $[pr, pl'] \supseteq [e', pl'] \supseteq [s', pl']$ holds.

Proof: Let p_c be the right child of p. p' can be either p_c or a descendant of p_c. By definition, e is in the subtree of p'. Hence e' is also in the subtree of p' and hence is also in the subtree of p_c. Since pr is p_c's leftmost child, it follows that $[pr, pl'] \supseteq [e', pl']$ holds. Clearly, $[e', pl'] \supseteq [s', pl']$ since $|[pr', e]| \geq |[s', pl']|$ and $|[pr', e]| = [pl', e']|$. ∎

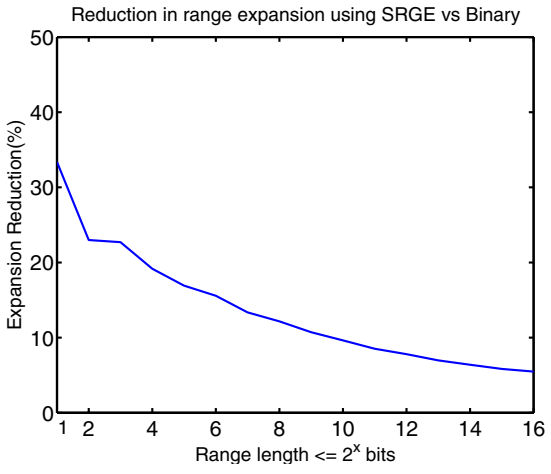

Fig. 6. Random database: reduction in range expansion achieved by SRGE as a function of range length

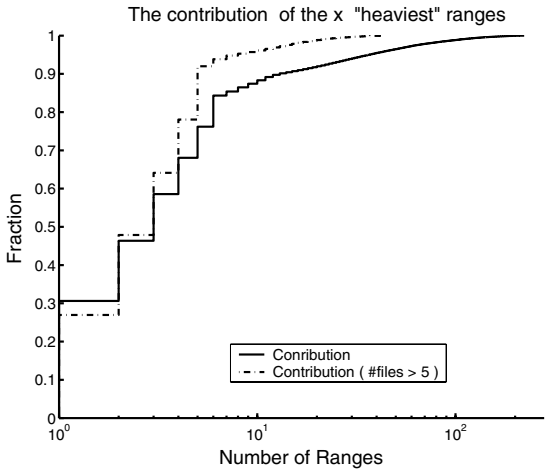

Fig. 7. Distribution of the contribution of the x heaviest ranges to the overall expansion

Claim 2: In case II: $[pr, pl'] \supseteq [ql, pl'] \supseteq [s', pl']$

Proof: Let p_c be the right child of p. p' can be either p_c or a descendant of p_c. $[pr, pl'] \supseteq [ql, pl']$ clearly holds since q is a prefix in the subtree of p' hence also in the subtree of p_c, and pr is the leftmost child in p_c. Showing that $[ql, pl'] \supseteq [s', pl']$ is also straightforward from the fact that s' is in the left subtree of p' and hence also in the subtree of q. ∎

Claim 3: A prefix in binary encoding translates to a prefix in BRGC.

The proof follows from the definition of BRGC by a simple induction on the number of bits. ∎

For a range $[e, s]$ we let n_{SRGE} denote the size of the set of ternary strings covering the range $[e, s]$ that results from applying the $SRGE$ algorithm. Let n_{prefix} denote the size of the set of prefixes by which the prefix expansion technique covers $[e, s]$.

Lemma 4: For all ranges $[e, s]$, $n_{SRGE} \leq n_{prefix}$ holds.

Sketch of Proof It can be easily shown that the prefix expansion algorithm must use the prefixes that cover the ranges $[s, pl]$ and $[pr', e]$ and, in case II, at least one additional prefix. The SRGE algorithm either uses these prefixes or, by using the reflection property of BRGC, coalesces pairs of these prefixes thus reducing the size of the covering set. ∎

Lemma 5: The worst case expansion of SRGE algorithm is $2W - 4$

Sketch of Proof: We prove the claim by a variation on the proof of the binary worst case expansion in [11] adopted to code

gray. Note, that it is easy to prove that the range $[1, 2^{W-1} + 2^{W-2} - 2]$ requires expansion $2W - 4$. ∎

Lemma 6: The time complexity of finding an SRGE cover is $O(W)$.

Sketch of Proof: The dominating factor in the time complexity of an SRGE cover construction is that of finding the prefix cover of two ranges: one in stage (3.1) and another either in stage (5.1) or in stage(6.1). It is easily seen that in both cases the time complexity is $O(W)$. ∎

The proof of the following lemma is omitted for lack of space.

Lemma 7: Let R be a BRGC range. Then the cover obtained by SRGE for R is optimal.

VIII. A Lower bound on Range Expansion Without Extra Bits

In this section we prove that any range-representation algorithm that does not use extra bits has a worst-case expansion ratio of at least w, regardless of the number encoding it uses.

A *w-encoding* is a 1:1 mapping of the integers in $\{0, \cdots, 2^w - 1\}$ onto the set of binary strings of length w. A string *matches* a set of ternary strings $\mathcal{P}$ if it matches at least one string of $\mathcal{P}$.

Lemma 8: Let $\mathcal{P}$ be a set of ternary strings. If exactly $2^w - 1$ w-bit numbers match $\mathcal{P}$ then $|\mathcal{P}| \geq \mathtt{w+1}$ holds.

Proof: The proof goes by induction. For $w = 1$, clearly, a set of ternary strings that is matched by either 0 or 1 must be of length at least 1. Assume the claim holds for $w = i$, we now prove for *w=i+1*. Let $\mathcal{P}$ be a set of ternary strings that is matched by exactly 2^{w+1}-1 *(w+1)*-bit binary numbers. Assume that $|\mathcal{P}| < \mathtt{w+1}$ holds to obtain a contradiction. Let $b_{w+1}b_w \cdots b_0$ be the single *(w+1)*-bit number that does not match $\mathcal{P}$. Consider the bits in position *w+1* of the strings in $\mathcal{P}$. If all these bits are in $\{b_{w+1}, *\}$, then either $(1-b_{w+1})b_w \cdots b_0$ does not match $\mathcal{P}$ or $b_{w+1}b_w \cdots b_0$ matches $\mathcal{P}$. Both these cases contradict our assumptions. Thus there must be at least one string in $\mathcal{P}$ whose most significant bit is $1 - b_{w+1}$. Let $\mathcal{P}'$ be the set obtained from $\mathcal{P}$ by removing all such strings from $\mathcal{P}$ and truncating the most significant bit of all remaining strings. Then $\mathcal{P}'$ covers all w-bit numbers except for $b_w \cdots b_0$ and is of length less than w. This is a contradiction. ∎

Lemma 9: The worst-case expansion ratio of any scheme is at least w, regardless of the encoding.

Proof: From Lemma 8, the range $[0, \cdots, 2^w - 2]$ cannot be represented by a prefixes set of size less than w. ∎

IX. Conclusions and Future Work

We have presented the SRGE algorithm for the efficient encoding of short ranges. The SRGE algorithm achieves a significant reduction in range expansion without resorting to the use of extra bits. We've also shown that the hybrid-SRGE scheme, that combined SRGE with the dependent-encoding of a small number of large high-expansion ranges, dramatically reduces the range expansion of a large real-life database from 2.7 to 1.03 as compared to prefix expansion, better than any prior art algorithms.

The hybrid-SRGE scheme is much more scalable than prior art. This is because small ranges, which constitute the majority of today's real-life classification databases, can be efficienty encoded by SRGE without using extra bits. SRGE also supports fast incremental updates since only a small number of known ranges are encoded using extra bits. Finally, the packet processing time required by SRGE is very small, since the transformation from a W-bit binary-encoded number to a W-bit BRGC number requires only W exclusive-or operations.

We have shown a lower bound of W on the worst-case expansion ratio of any ternary encoding scheme. The SRGE algorithm achieves worst-case expansion of $2W - 4$. We conjecture that there exist ternary encoding schemes with better worst-case expansion ratio. Finding the tight bound on the worst-case expansion ratio of ternary encoding schemes remains an interesting open problem. We note, however, that lower worst-case expansion ratio does not necessarily imply lower *average* expansion ratio.

X. Acknowledgment

The authors are deeply indebted to Cisco Systems, Will Eatherton, and David Taylor for kindly providing us the classification database we used for this work. We would also like to thank Yehuda Afek, Karthik Lakshminarayanan, Anand Rangarajan , and Srinivasan Venkatachary for assisting us in the process of obtaining access to the database. We are also indebted to Ronny Roth for pointing out the connection between the TCAM range encoding and DNF expression minimization problems and for helpful discussions.

References

[1] D.E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computer Surverys*, pp. 238–275, 2005.

[2] Pankaj Gupta and Nick McKeown, "Algorithms for packet classification," in *IEEE Network Special Issue*, 2001.

[3] George Varghese, *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*, The Morgan Kaufmann Series in Networking, 2005.

[4] Florin Baboescu, Sumeet Singh, and George Varghese, "Packet classification for core routers: Is there an alternative to cams," in *INFOCOM*, 2003.

[5] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," in *ACM SIGCOMM 98*, Sept. 1998, pp. 191–202.

[6] H. Liu, "Efficient mapping of range classifier into ternary-cam," in *Hot Interconnects*, 2002.

[7] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *JSAC*, 2003.

[8] Srinivasan Venkatachary Karthik Lakshminarayanan, Anand Rangarajan, "Algorithms for advanced packet classification with ternary cams," in *SIGCOMM*, 2005.

[9] F.Yu and R.H. Katz, "Efficient multi-match packet classification with tcam," in *HOTI*, 2004.

[10] D. Taylor E.Spitznagel and J. Turner, "Packet classification using extended tcams," in *ICNP*, 2003.

[11] Baruch Schieber, Danny Geist, and Ayal Zaks, "Computing the minimum dnf representation of boolean functions defined by intervals," *Discrete Applied Mathematics*, , no. 1-3, pp. 154–173, 2005.

[12] "Ports numbers," 2006, http://www.iana.org/assignments/port-numbers.

[13] Jonathan S. Turner David E. Taylor, "Classbench: A packet classification benchmark," in *IEEE INFOCOM*, 2005.