Layered Interval Codes for TCAM-based Classification*

Anat Bremler-Barr
Interdisciplinary Center
Herzliya
bremler@idc.ac.il

David Hay
Ben-Gurion University
davidhay@cs.bgu.ac.il

Danny Hendler
Ben-Gurion University
hendlerd@cs.bgu.ac.il

Boris Farber
Interdisciplinary Center
Herzliya
farber.boris@idc.ac.il

Categories and Subject Descriptors: C.2.6 Internetworking: Routers
General Terms: Algorithms
Keywords: TCAM,Classification

1. INTRODUCTION

Packet classification is an indispensable building block of numerous Internet applications in the areas of routing, monitoring, security, and multimedia. The routers use a *classification database* that consists of a set of *rule*s (a.k.a. *filter*s). Each such rule specifies a pattern that determines which packets will match it and which actions to apply to the matched packets.

Ternary content-addressable memories (TCAMs) are increasingly used for high-speed packet classification. A TCAM is an associative memory hardware device that can be viewed as an array of fixed-width entries. Each TCAM entry consists of ternary digits: 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. TCAMs are not well-suited, however, for representing rules that contain range fields (such as port fields). The traditional technique for range representation is the *prefix expansion* technique [6], in which a range is represented by a set of prefixes, such that each is stored in a single TCAM entry. For example, the range $[1, 6]$ can be represented by the prefix set $\{001, 01*, 10*, 110\}$. Hence, a single rule may require multiple TCAM entries, resulting in *range expansion*. Range expansion was found to cause an increase of more than 16% in TCAM space requirements for real-word databases [7].

We present a novel scheme for constructing efficient representations of range rules, by making use of extra bits that are available in each TCAM entry. Since TCAMs are typically configured to be 144 bits wide, a few dozens of unused bits, called *extra bits*, remain in each entry. Our scheme is based on the simple observation that sets of disjoint ranges may be encoded much more efficiently than sets of overlapping ranges. Since the ranges in real-world classification databases are, in general, non-disjoint, our technique splits the ranges between multiple *layers*, each of which consists of mutually disjoint ranges. Each layer is then coded independently and assigned its own set of extra bits. We call the resulting encoding scheme a *Layered Interval Code* (LIC).

We present algorithms that implement the proposed scheme, where the *layering* of ranges is done by approximation algorithms for specific variants of interval-graph coloring. In addition, in the full paper [1] we provide mechanisms for updating TCAM entries without interfering with ongoing TCAM searches (*hot updates*).

We evaluate these algorithms by performing extensive comparative analysis on real-life classification databases. Our analysis establishes that our algorithms reduce the number of redundant TCAM entries caused by range rules by more than 60% as compared with best range-encoding prior art.

On the theoretical side, we prove in the full paper that constructing a LIC that uses a minimum number of bits is NP hard. We also prove that a polynomial-time 2-approximation algorithm for the closely related *chromatic sum* problem is also a 2-approximation algorithm for the LIC construction problem. We then prove that the problem of constructing an optimal LIC given a *a specific budget of extra bits* is also NP hard.

2. BACKGROUND

Prior art that deals with reducing range expansion can roughly be divided to two main categories. *Database-dependent schemes* may encode a range differently as a function of the distribution of ranges in the database in which it occurs. The encoding produced by *database-independent schemes*, on the other hand, is a function of the range only and does not change across different databases [2, 4]. The scheme we present here is database-dependent.

The first database-dependent prior art is due to Liu [5]. It uses the available extra bits as a bit map: a single extra bit is assigned to each selected range r (see Figure 1(a)). If range r is assigned extra bit i, then the i'th extra bit is set in all TCAM entries that include r; all other extra bits are set to 'don't care' in these entries. In the search key, extra bit i is set to 1 if the key falls into range r, otherwise it is set to 0. Since there are typically only 36 available extra bits per entry, this scheme is not scalable. Moreover, the number of unique ranges in contemporary classification databases is around 300 [4] and is expected to grow, thus this scheme may already be insufficient.

Two algorithms - *Region Partitioning* [5] and *DRES* [3] - were proposed for alleviating this scalability problem. DRES is a greedy algorithm that assigns extra bits to the ranges with high prefix expansion. The key difference between DRES and our scheme is that, whereas DRES assigns a single bit per range, we use the much more compact LIC coding. An idea similar to our LIC technique was mentioned in [8] but no algorithm was presented, hence also no empirical data was shown.

*This work was partly supported by a Cisco grant.

SIGMETRICS'08, June 2–6, 2008, Annapolis, Maryland, USA.
ACM 978-1-60558-005-0/08/06.

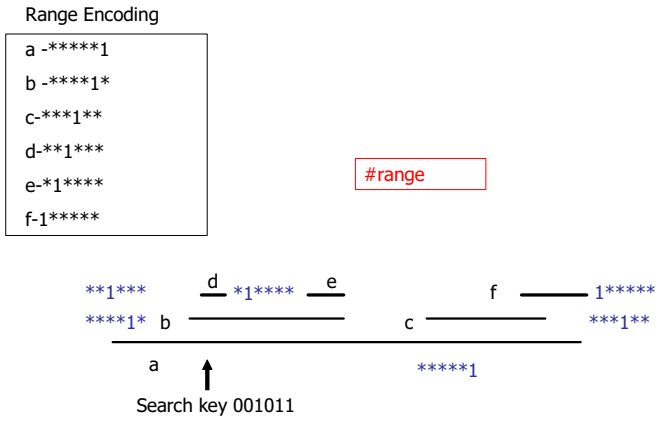

Figure 1(a): Liu's Basic Dependent Encoding

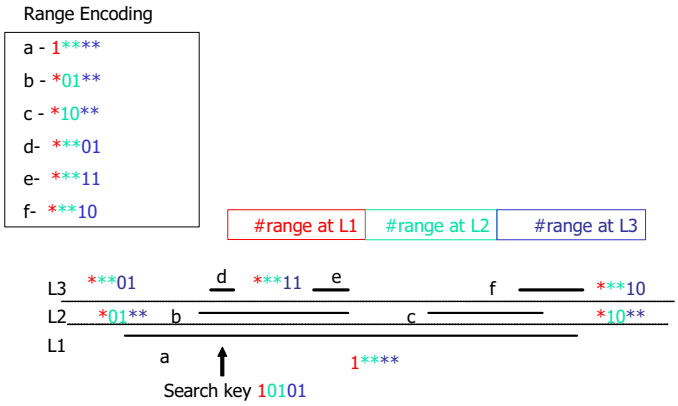

Figure 1(b): Example of the LIC encoding

3. ALGORITHM

The main observation on which the LIC algorithm is based is the fact that, while encoding n arbitrary ranges may require n bits, only $\log(n+1)$ bits are required to encode n *disjoint* ranges. This is because a search key will fall into at most one of these ranges.

Our algorithm is composed of three stages: a *layering stage*, a *bit allocation stage*, and an *encoding stage*. In the *layering stage*, we partition the ranges to *layers*, where each layer contains a disjoint set of ranges. We propose and evaluate four polynomial-time layering algorithms: **a) Maximum Size Independent Sets (MSIS)** - a greedy layering algorithm that iteratively finds maximum independent sets, **b) Maximum Size Colorable Sets (MSCS)** - this algorithm finds, in every iteration i, the maximum i-colorable set of the interval graph corresponding to the ranges set. **c) Maximum Weight Independent Sets (MWIS)**, and **d) Maximum Weight Colorable Sets (MWCS)** are weighted versions of the former algorithms.

In the *bits allocation stage*, we choose the "heaviest" ranges of each layer to be encoded, minimizing range-expansion while not exceeding each layer's extra bits budget.

Finally, the *encoding stage* constructs corresponding search keys and entries. A search key is constructed by encoding, for each layer, the single range of this layer to which the search key falls; if no such range exists, we encode a value representing the "area" outside all of this layer's ranges. A range-entry is constructed by encoding the range (at the single layer to which it belongs), while using 'don't care' bits for all other layers. Figure 2 (b) illustrates LIC encoding. For algorithm pseudo-codes and detailed descriptions, please refer to the full paper [1].

4. EXPERIMENTS

We measure the quality of a range encoding scheme E by the *expansion factor* and *range redundancy factor* metrics. The expansion factor is the relative increase in the number of entries required to represent database $\mathcal{D}$ in TCAM by using encoding scheme E. Since this paper deals with range-field encoding, we focus our attention on rules containing such fields: The *range redundancy factor* of database $\mathcal{D}$, using scheme E, is the average redundancy over all range-rules $R \in \mathcal{D}$, where the redundancy of a range R is defined as the expansion of R minus one.

We evaluate the performance of the LIC scheme on a real-life database, which is a collection of 120 separate rule files originating from various applications (such as firewalls, acl-routers, and intrusion prevention systems) collected over the year 2004.

Algorithm	Expansion	Redundancy
Prefix Expansion [6]	2.68	6.53
Region Partitioning [5]	1.64	2.51
hybrid DIRPE [4]	1.2	-
hybrid SRGE [2]	1.03	1.2
DRES [3]	1.025	0.09
LIC/MSIS and SRGE	1.0061	0.024
LIC/MSCS and SRGE	1.0075	0.029

Table 1: Expansion and redundancy factors, using 36 extra bits, for different range encoding algorithms.

Our database consists of a total of approximately $223,000$ rules that contain 280 unique ranges. Approximately 28% of the rules contain range fields and about half of these include the range $[1024, 2^{16}-1]$.

Table 1 shows the expansion and redundancy factors obtained by prior art and by our algorithms, when 36 extra bits are available. When all extra bits are exhausted, our LIC scheme can use any independent encoding algorithm as a fall-back scheme. We evaluated the LIC scheme using two fall-back schemes: (binary) prefix expansion and SRGE (binary-reflected Gray code) [2]. The redundancy factor obtained by all LIC algorithms reduces redundancy by 60% or more as compared with the best prior art algorithm.

5. REFERENCES

[1] A. Bremler-Barr, D.Hay, D. Hendler, and B. Ferber. Layered interval codes, 2008. http://www.idc.ac.il/staff/publicationAbstract.asp?publicationID=1631.

[2] A. Bremler-Barr and D. Hendler. Space-efficient tcam-based classification using gray coding. In *IEEE INFOCOM*, pages 1388–1396, 2007.

[3] H. Che, Z. Wang, K. Zheng, and B. Liu. Dres: Dynamic range encoding scheme for tcam coprocessors. *IEEE Transaction on Computers*, 2008. To appear.

[4] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with ternary CAMs. In *ACM SIGCOMM*, 2005.

[5] H. Liu. Efficient mapping of range classifier into ternary-cam. In *Hot Interconnects*, 2002.

[6] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *ACM SIGCOMM*, pages 191–202, Sept. 1998.

[7] D. E. Taylor. Survey and taxonomy of packet classification techniques. *ACM Comput. Surv.*, 37(3):238–275, 2005.

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