Compressing Forwarding Tables

Ori Rottenstreich*, Marat Radan*, Yuval Cassuto*, Isaac Keslassy*,
Carmi Arad§, Tal Mizrahi§*, Yoram Revah§ and Avinatan Hassidim¶

*Technion, {or@tx, radan@tx, ycassuto@ee, isaac@ee}.technion.ac.il
§Marvell Israel, {carmi, talmi, yoramr}@marvell.com
¶Google Israel and Bar-Ilan University, avinatanh@gmail.com

***Abstract*—With the rise of datacenter virtualization, the number of entries in forwarding tables is expected to scale from several thousands to several millions. Unfortunately, such forwarding table sizes can hardly be implemented today in on-chip memory.**

In this paper, we investigate the compressibility of forwarding tables. We first introduce a novel forwarding table architecture with separate encoding in each column. It is designed to keep supporting fast random accesses and fixed-width memory words. Then, we suggest an encoding whose memory requirement per row entry is guaranteed to be within a small additive constant of the optimum. Next, we analyze the common case of two-column forwarding tables, and show that such tables can be presented as bipartite graphs. We deduce graph-theoretical bounds on the encoding size. We also introduce an algorithm for optimal conditional encoding of the second column given an encoding of the first one. In addition, we explain how our architecture can handle table updates. Last, we evaluate our suggested encoding techniques on synthetic forwarding tables as well as on real-life tables.

I. Introduction

A. Background

The rapid growth of forwarding tables in network switches raises serious concerns in the networking industry. Layer 2 (L2) networks are no longer constrained to small local area networks. On the contrary, they are now used in datacenter networks, which will soon have a need for millions of hosts in a single L2 domain, while current L2 domains are only restricted to several thousands [1]. As a result, current forwarding tables cannot handle such datacenter networks.

The main driver of this expected dramatic growth rate is the deployment of host *virtualization* in datacenter networks. Whereas traditional servers would only use one or two globally-unique MAC addresses per server, contemporary servers use tens of MAC addresses, corresponding to their tens of virtual machines (these generated MAC addresses are not necessarily globally unique anymore). Assuming one MAC address per thread, and considering the joint increase of the numbers of (a) threads per core, (b) cores per processor (socket), (c) sockets per server, and (d) servers per datacenter, it is clear that the product of all these numbers is expected to dramatically increase in the coming years [1]–[3]. In addition to this expected strong growth in the number of entries per forwarding table, we can also expect a moderate growth in the number of attributes per entry, such as the quality-of-service and security attributes, as well as in the size of these attributes. All in all, we need to be able to handle dramatically larger forwarding tables.

Unfortunately, forwarding tables must be accessed on a per-packet basis. Therefore, they are performance sensitive, and are often implemented in hardware using *on-chip memory*. This makes scaling forwarding tables challenging, because on-chip memory is too small to hold millions of forwarding-table entries in raw form [4], [5].

The goal of this paper is to study the compressibility of forwarding tables, providing both an analytical framework and experimental insight toward the effective compression of such tables.

B. Compressing Forwarding Tables

Forwarding tables, also referred to as Forwarding Information Bases (FIBs), consist of several entries, which associate a *lookup key* to its corresponding *forwarding information*. For instance, the lookup key can be a (*VLAN*, *destination MAC address*) pair, and the forwarding information can include the switch target port, the quality-of-service attributes, the security attributes, and additional information used for various modifications that need to be applied to the packet.

Forwarding tables are typically arranged as *two-dimensional tables*. Each row entry in a forwarding table corresponds to a different lookup key, and contains several columns, with the lookup key and forwarding information. Forwarding tables can then be accessed using a hashing mechanism. Upon each incoming packet, the packet lookup key is hashed into one (or several) of the hash table entries. If this entry contains indeed the lookup key, then the forwarding table returns the associated forwarding information.

Forwarding tables are typically redundant, because their fields (columns) can have a highly non-uniform value distribution, and because the values in different fields can be correlated. Therefore, we are interested in studying whether these properties can be used to compress them. However, we would like to keep two key properties that are specific to forwarding tables:
(i) *The ability to access each row entry directly regardless of its row index*, and
(ii) *The ability to store each row entry in a fixed-width memory word.*

Such criteria prevent or make less attractive traditional encoding techniques. For instance, if the entries are compressed using a Lempel-ZivWelch (LZW) technique and sequentially stored in the memory, it is then difficult to calculate the memory

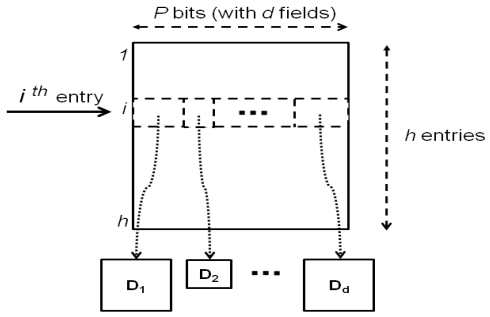


Fig. 1. An illustration of an access to the encoded forwarding table. First, a single encoded entry of (at most) P bits is retrieved. Then, d pipelined accesses to d dictionaries are performed based on the d binary codewords (of different lengths) composing this encoded entry.

address of a specific entry, and to access it directly based on its entry index.

C. Overview of the Suggested Solution

As illustrated in Figure 1, we suggest a *novel compression technique for forwarding tables* that supports random accesses.

We propose to encode each of the forwarding-table columns separately, using a dedicated variable-length *binary prefix encoding*, i.e. an encoding in which any codeword is not a prefix of any other codeword. Therefore, each row entry in a d-column table is represented by the concatenation of d codewords representing the content of each of its d fields. Accordingly, its width equals the sum of the d corresponding codeword lengths. For each row entry we allocate a fixed memory size P. In addition, each column i is associated to a dictionary D_i, which is stored separately in the memory. This dictionary D_i maps each binary codeword into its represented field value in column i.

Figure 1 illustrates an access to the encoded forwarding table. At each packet arrival, we first retrieve the P bits representing the encoding of its i^{th} entry. Then, we look up each of the d dictionaries in a pipelined manner. This way, we sequentially obtain the corresponding values of the d fields in the i^{th} entry after d pipelined lookups.

Note that the combination of the bounded-width entry property and the prefix property of individual fields yields our two desired properties specified above.

Example 1. *Table I presents an example of a forwarding table and a possible encoding of that table. There are $h = 7$ entries in the table, each with $d = 3$ fields: Target Port, MAC Address and VLAN.*

First, Table (A) presents the original table without encoding.

Next, Table (B) illustrates a possible encoding of the forwarding table (the presented spaces in the encoded table are just shown to distinguish between the different fields, and do not exist in practice). For instance, the two possible values in the VLAN field are encoded by a simple fixed-length single-bit encoding. In addition, variable-length encodings are used to encode the three distinct values in the Target Port field, as well as the five distinct values of the MAC Address field. With these $d = 3$ encodings, the obtained maximal width of an encoded entry is $P = 5$ bits. Thus, in order to support the random-access requirement, we keep $P = 5$ bits in the memory for each encoded entry, even if it may actually be shorter. For instance, the third entry only needs 4 bits, and therefore wastes 1 bit.

Finally, Table (C) illustrates the encodings used by the $d = 3$ dictionaries representing the $d = 3$ columns. The number of entries in each dictionary equals the number of distinct values in the corresponding column in (A).

(A) The original forwarding table

Target Port	MAC Address	VLAN
Te12/1	00:1b:2b:c3:4d:90	Vlan10
Gi11/8	00:00:aa:6c:b1:10	Vlan10
Te12/1	00:00:aa:65:ce:e4	Vlan10
Gi11/24	00:00:aa:65:ce:e4	Vlan200
Gi11/24	00:13:72:a2:a2:0e	Vlan200
Te12/1	00:21:9b:37:7e:14	Vlan10
Gi11/8	00:13:72:a2:a2:0e	Vlan200

(B) Encoded table

0 110 0
10 00 0
0 01 0
11 01 1
11 10 1
0 111 0
10 10 1

(C) The encoding dictionaries

D_1 (Target Port)	D_2 (MAC Address)	D_3 (VLAN)
0 - Te12/1	00 - 00:00:aa:6c:b1:10	0 - Vlan10
10 - Gi11/8	01 - 00:00:aa:65:ce:e4	1 - Vlan200
11 - Gi11/24	10 - 00:13:72:a2:a2:0e	
	110 - 00:1b:2b:c3:4d:90	
	111 - 00:21:9b:37:7e:14	

TABLE I

AN EXAMPLE OF A FORWARDING TABLE WITH $d = 3$ COLUMNS AND ITS ENCODED REPRESENTATION. (A) SHOWS THE ORIGINAL TABLE WITHOUT ENCODING. THE ENTRY SELECTION MECHANISM IS NOT PRESENTED. (B) ILLUSTRATES THE CORRESPONDING ENCODED FORWARDING TABLE WITH A MAXIMAL ENCODED ENTRY WIDTH OF $P = 5$ BITS. (SPACES ARE PRESENTED JUST FOR SIMPLICITY AND DO NOT EXIST IN PRACTICE.) (C) DESCRIBES THE $d = 3$ DICTIONARIES REPRESENTING THE PREFIX ENCODINGS OF THE d COLUMNS.

This example illustrates how prefix encoding enables us to simply *concatenate the codewords* without a need for any additional separating flags.

In addition, it also shows how the memory size is directly determined by the *worst-case entry width*, i.e. the maximum number of concatenated bits per entry. This is one reason for example why encoding each column using *Huffman coding* [6] would not necessarily address this problem: while it is optimal for the *average* entry width, it may yield a worst case that ends up wasting a significant number of bits. Therefore, for a given forwarding table with d columns, *our goal is to suggest a set of d prefix encodings that minimizes the maximal encoded entry width.* This makes it particularly hard, because each entry width involves *all columns*, and the maximal one also involves *all row entries*. Therefore, any strong solution may need to *jointly* look at all the elements.

D. Related Work

The forwarding table scaling problem has been thoroughly analyzed and discussed in networking industry forums and standards organizations, both in the L2 and the L3 layers (e.g. [1], [3]–[5], [7]). These analyses raise the concern that

the steep growth rate of forwarding tables has outrun the steady technology growth rate of memory density.

Various creative methods have been proposed to mitigate this scaling problem. First, *FIB aggregation* [8] is an approach that reduces the size of a forwarding table by aggregating table entries with identical forwarding information, i.e identical entries without the lookup part. However, this approach does not allow efficient representation when FIB entries are highly correlated but not identical. This is increasingly the case, as the number of columns per entry grows, and therefore the potential for two row entries to be different in at least one column grows as well. Similar solutions can be found in [9], [10].

Another approach to scaling forwarding tables is by distributing the forwarding table among multiple nodes in the network. An example of this approach is *FIB suppression with virtual aggregation* [11], which distributes the L3 forwarding table among several routers, thus allowing each router to maintain a small subset of the full address set. Distributed table approaches yield a management overhead and are typically applicable to L3 address tables, but are also less relevant to L2 tables.

In addition, there have been studies in other related fields. Clearly, in *information theory*, there are many results on compression and on lower bounds based on entropy [6]. Likewise, *database implementations* also attempt to scale systems by introducing compression techniques [12]–[14]. However, these studies do not consider the forwarding table restrictions that we mention above, and in particular the bounded entry length.

Finally, as mentioned, an orthogonal and complementary consideration is the implementation of the forwarding table using hashing, in order to easily find the row that corresponds to a given key [15]–[17].

E. Contributions

This paper investigates a novel technique for efficient representation of forwarding tables, with a support for fast random access and fixed-width memory words.

We start by studying an offline encoding scheme for a given static FIB table. We prove that it is possible to formulate the encoding problem as an optimization problem, and that the relaxed optimization problem is convex. As a result, we propose a novel algorithm that achieves the optimal worst-case entry width *within a guaranteed constant additive factor of* $d-1$ in any forwarding table with d columns, independently of their values and of the number of rows.

In addition, we also consider the special case of forwarding tables with $d = 2$ columns, which may only consist of the key and the action. We first suggest tight upper and lower bounds on the optimal encoding width of a two-column table in the general case. We also show a sufficient condition for a fixed-length encoding to be optimal. Moreover, we present an *optimal conditional encoding* of a second column when the encoding of one column is given. Later, we also introduce a bipartite graph that can model the forwarding table, and present additional graph-theoretic bounds on its optimal encoding width. These improved bounds make use of combinatorial properties of the graph, such as the size of its edge cover.

Finally, we also discuss the limits of our offline model, by analyzing how our compressed forwarding tables may be able to handle *updates*. We show that an additional bit per column and an additional small memory may be sufficient to cope with temporary loads of entry additions, modifications or deletions.

We conclude by evaluating the suggested encoding techniques on real-life forwarding tables as well as on synthetic tables. We demonstrate how our introduced techniques fare against alternative algorithms such as fixed-width and Huffman coding. In particular, we get closer to the compression lower bound, albeit at the cost of an increased complexity.

Due to space limitations, some of the proofs in this paper are deferred to [18].

II. Model and Problem Formulation

A. Terminology

We start with the formal definition of terminology used in this paper. For a binary codeword (string) x, let $\ell(x)$ denote the length in bits of x.

Definition 1 (FIB Table)**.** *A* FIB table $A = ((a_1^1, \ldots, a_d^1), \ldots, (a_1^h, \ldots, a_d^h))$ *is a two-dimensional matrix of h rows (entries) and d columns (fields).*

Given a FIB table A, we denote by S_j (for $j \in [1, d]$) the set of element values in the j^{th} column, i.e. $S_j = \bigcup_{i=1}^{h} \{a_j^i\}$. We also denote by n_j the number of distinct elements in the j^{th} column, i.e. $n_j = |S_j|$. Let $s_{j,1}, \ldots, s_{j,n_j}$ denote the elements of S_j.

For example, in Table I, there are $n_1 = 3$ values in the first column, and $S_1 = \{$Te12/1,Gi11/8,Gi11/24$\}$.

Definition 2 (Prefix Encoding)**.** *For a set of elements S, an encoding σ is an injective mapping $\sigma : S \to B$, where B is a set of binary codewords of size $|B| = |S|$. An encoding is called a* prefix encoding *if no binary codeword in B is a prefix (start) of any other binary codeword in B.*

Definition 3 (FIB Encoding Scheme)**.** *An* encoding scheme *C_A of a FIB Table A is an ordered set of d* prefix *encodings $C_A = (\sigma_1, \ldots, \sigma_d)$. That is, each σ_j encodes column j, and is constrained to be a prefix encoding.*

Definition 4 (FIB Encoding Width)**.** *Let $C_A = (\sigma_1, \ldots, \sigma_d)$ be a FIB encoding scheme of the FIB table A. The encoding width $\ell(C_A)$ of C_A is defined as the maximal sum of the lengths of the d codewords representing the d elements in a row of A, i.e.*

$$\ell(C_A) = \max_{i \in [1,h]} \left(\sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \right). \tag{1}$$

Example 2. *Let A be the FIB table from Example 1 with $h = 7$, $d = 3$, also described in Table I.(A). Likewise, let $C_A = (\sigma_1, \sigma_2, \sigma_3)$ be the FIB encoding scheme of A. Then, as shown in Table I.(C), σ_1(Te12/1) = 0, σ_1(Gi11/8) = 10, σ_1(Gi11/24) = 11. As described in Table I.(B), $\ell(C_A) = \max_{i \in [1,h]} \left(\sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \right) = \max \left(5, 5, 4, 5, 5, 5, 5 \right) = 5$.*

Let's compare this scheme with a fixed-length encoding of each column. Since the number of distinct elements in the $d = 3$ columns of A are $n_1 = 3, n_2 = 5$ and $n_3 = 2$, if I_A a fixed-length encoding of A then the elements in the j^{th} column are encoded in $\lceil \log_2 n_j \rceil$ bits, and $\ell(I_A) = \sum_{j=1}^{d} \lceil \log_2 n_j \rceil = 2 + 3 + 1 = 6$.

B. Optimal FIB Table Encoding Scheme

For each FIB table A, we denote by $OPT(A)$ the optimal encoding width of A, i.e. the smallest possible encoding width of any encoding scheme of A such that

$$OPT(A) = \min_{C_A = (\sigma_1, \ldots, \sigma_d)} \ell(C_A). \tag{2}$$

Our goal is to find an encoding scheme C_A that minimizes the encoding width $\ell(C_A)$, and therefore reaches this minimum.

III. General FIB Tables

A. Optimization Problem

We first want to rewrite the problem of finding the optimal encoding scheme for any FIB table A as an optimization problem.

First, it is well known that the set of codeword lengths in a prefix encoding exists iff it satisfies *Kraft's inequality* [6].

Property 1 (Kraft's inequality)**.** *There exists a prefix encoding σ of the elements in a set S with codeword lengths $\{\ell(\sigma(a)) | a \in S\}$ iff*

$$\sum_{a \in S} 2^{-\ell(\sigma(a))} \leq 1. \tag{3}$$

Therefore, we can now formally present the problem of finding an optimal FIB table encoding scheme for table $A = ((a_1^1, \ldots, a_d^1), \ldots, (a_1^h, \ldots, a_d^h))$ as the following optimization problem.

$$\begin{aligned}
\min \quad & P \\
\text{s.t.} \quad & \sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \leq P && \forall i \in [1, h] && \text{(4a)} \\
& \sum_{a \in S_j} 2^{-\ell(\sigma_j(a))} \leq 1 && \forall j \in [1, d] && \text{(4b)} \\
& \ell(\sigma_j(a)) \geq 0 \ \forall j \in [1, d], && \forall a \in S_j && \text{(4c)} \\
& \ell(\sigma_j(a)) \in \mathbb{Z} \ \forall j \in [1, d], && \forall a \in S_j && \text{(4d)}
\end{aligned}$$

In this optimization problem, we try to minimize the maximal encoding width of a row (denoted here by P) while having four sets of constraints. The first set (4a) represents the limitation on the total width of each row. The second set of constraints (4b) requires that each of the encodings $\sigma_1, \ldots, \sigma_d$ should satisfy Kraft's inequality. The last two sets (4c, 4d) illustrate the fact that the codeword length of any element should be a non-negative integer (the constraints (4b) and (4c) together guarantee that if the number of distinct elements in a column is at least two, the codeword lengths are positive). For an optimal solution to this optimization problem, the equality $P = OPT(A)$ is satisfied.

The optimization problem can be also described as being dependent only on the codeword lengths. To do so, we replace (for $j \in [1, d], a \in S_j$) the value of $\ell(\sigma_j(a))$ by a corresponding variable. In addition, as explained in [18], it is easy to derive the sets of codewords given their lengths.

We also denote by the *relaxed FIB encoding problem* the problem achieved by omitting the fourth set of constraints (4d). In this relaxed problem, the codeword lengths are not necessarily integers.

B. Approximation of the Optimal Encoding

In [18], we prove that finding the optimal encoding of a given FIB table is NP-hard in the general case. We now show how to find for each FIB table A of d columns a solution to the FIB encoding problem that is guaranteed to be within a *fixed additive approximation* of the optimal encoding width $OPT(A)$. More specifically, we find an encoding scheme C_A with encoding width $\ell(C_A)$ that satisfies $\ell(C_A) \leq OPT(A) + (d - 1)$, i.e. its encoding width is larger than the optimal encoding width (in bits) by at most the number of columns in A minus one. We emphasize that this bound on the additive error of $(d - 1)$ depends neither on the number of distinct elements $n_j \leq 2^{W_j}$ (for $j \in [1, d]$) in each of the columns, nor on the number of rows h in A.

To obtain such an encoding, we consider the *relaxed FIB encoding problem* defined above, in which the codeword lengths are not necessarily integers. We show that this optimization problem is *convex*, and thus its solution can be found by one of several known algorithms for such problems. We then build as a solution to the original optimization problem, an encoding with codeword lengths achieved by taking the ceiling of each of the codeword lengths in the solution of the relaxed problem. We show that these new codeword lengths lead to an encoding that satisfies the additive approximation from above.

Theorem 1. *Let $\left(\ell(\bar{\sigma}_j(s_{j,1})), \ldots, \ell(\bar{\sigma}_j(s_{j,n_j}))\right)$ be the codeword lengths of an optimal solution to the relaxed FIB encoding problem. Further, let $C_A = (\sigma_1, \ldots, \sigma_d)$ be an encoding scheme satisfying $\ell(\sigma_j(s_{j,i})) = \lceil \ell(\bar{\sigma}_j(s_{j,i})) \rceil$ for all $j \in [1, d], i \in [1, n_j]$. Then we have:*
(i) *The relaxed FIB encoding problem is convex.*
(ii) *The encoding width $\ell(C_A)$ of the encoding scheme C_A satisfies*

$$\ell(C_A) \leq OPT(A) + (d - 1). \tag{5}$$

Proof: We first examine the convexity of the relaxed problem. Clearly, its simple objective function is convex. We would like to show that each of the inequality constraint functions (as a function of the codeword lengths) is convex as well. Simply, constraints in the first and the third sets of inequality constraints (4a, 4c) are convex due to their linearity. In addition, we would now like to examine the convexity of the second set of constraints (4b) representing Kraft's inequality. To do so, we define d functions $f_1, \ldots, f_d$ for the constraints on

each of the d prefix encodings. The function f_j (for $j \in [1,d]$) receives as a parameter the set of codeword lengths of the elements in S_j. The function $f_j\big(\ell(\sigma_j(s_{j,1})),\ldots,\ell(\sigma_j(s_{j,n_j}))\big)$ is defined as $\sum_{a\in S_j} 2^{-\ell(\sigma_j(a))} = \sum_{i=1}^{n_j} 2^{-\ell(\sigma_j(s_{j,i}))}$.

We consider two arbitrary encoding schemes $(\bar{\sigma}_1,\ldots,\bar{\sigma}_d),(\hat{\sigma}_1,\ldots,\hat{\sigma}_d)$ with codeword lengths $\big(\ell(\bar{\sigma}_j(s_{j,1})),\ldots,\ell(\bar{\sigma}_j(s_{j,n_j}))\big)$, $\big(\ell(\hat{\sigma}_j(s_{j,1})),\ldots,\ell(\hat{\sigma}_j(s_{j,n_j}))\big)$ for the elements in S_j for $j \in [1,d]$, respectively. We would like to show that $f_j(\alpha\cdot\bar{\sigma}+\beta\cdot\hat{\sigma}) \le \alpha\cdot f_j(\bar{\sigma})+\beta\cdot f_j(\hat{\sigma})$ for all $j\in[1,d]$ and $\alpha,\beta\in[0,1]$ with $\alpha+\beta=1$. Here, when performing linear operations on the set of codeword lengths, we refer to a new set of codeword lengths, so that each of its codeword lengths is obtained by performing the linear operations on the corresponding length in the original set. Based on the convexity of the function $g(x)=2^{-x}$, we have

$$\begin{aligned} f_j(\alpha\cdot\bar{\sigma}+\beta\cdot\hat{\sigma}) &= \sum_{i=1}^{n_j} 2^{-\left(\alpha\cdot\ell(\bar{\sigma}_j(s_{j,i}))+\beta\cdot\ell(\hat{\sigma}_j(s_{j,i}))\right)} \\ &\le \sum_{i=1}^{n_j}\left(\alpha\cdot 2^{-\ell(\bar{\sigma}_j(s_{j,i}))}+\beta\cdot 2^{-\ell(\hat{\sigma}_j(s_{j,i}))}\right) \\ &= \alpha\cdot\sum_{i=1}^{n_j} 2^{-\ell(\bar{\sigma}_j(s_{j,i}))}+\beta\cdot\sum_{i=1}^{n_j} 2^{-\ell(\hat{\sigma}_j(s_{j,i}))} \\ &= \alpha\cdot f_j(\bar{\sigma})+\beta\cdot f_j(\hat{\sigma}). \end{aligned} \tag{6}$$

Based on these properties, we can finally deduce that the relaxed FIB encoding problem is convex. With this observation, an optimal solution, i.e. a generalized encoding (with non-necessarily integer codeword lengths) can be found by using one of the several known (polynomial time) algorithms for such convex problems (e.g. the ellipsoid algorithm [19]). Let us denote by

$$\big(\ell(\bar{\sigma}_j(s_{j,1})),\ldots,\ell(\bar{\sigma}_j(s_{j,n_j}))\big)$$

the codeword lengths of an optimal solution to the relaxed problem. Let $OPT_r(A) = \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\bar{\sigma}_j(a_j^i))\right)$ be the maximal encoding width of a row in this optimal solution of the relaxed problem. Again, $OPT_r(A)$ is not necessarily integer. Clearly, since the set of constraints in the relaxed problem is a subset of the constraints in the FIB encoding problem, the inequality $OPT_r(A) \le OPT(A)$ holds. Further, since $OPT(A)$ is an integer, we have that $\lceil OPT_r(A)\rceil \le OPT(A)$, i.e. $\lceil OPT_r(A)\rceil$ *is a lower bound for* $OPT(A)$. Clearly, since $\ell(\sigma_j(s_{j,i})) = \lceil\ell(\bar{\sigma}_j(s_{j,i}))\rceil \ge \ell(\bar{\sigma}_j(s_{j,i}))$, all the d constraints representing the Kraft's inequality, satisfied by the solution of the relaxed problem, are also satisfied by the set of codeword lengths in $C_A=(\sigma_1,\ldots,\sigma_d)$.

We now show that the encoding width of the encoding scheme C_A is within a guaranteed additive approximation of the optimal encoding width of A, $OPT(A)$. Directly from the definition of C_A, we can have that

$$\begin{aligned} \ell(C_A) &= \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\sigma_j(a_j^i))\right) = \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\lceil\ell(\bar{\sigma}_j(a_j^i))\rceil\right) \\ &< \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\left(\ell(\bar{\sigma}_j(a_j^i))+1\right)\right) \\ &= \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\bar{\sigma}_j(a_j^i))\right)+d = OPT_r(A)+d \\ &\le OPT(A)+d. \end{aligned} \tag{7}$$

Finally, since $\ell(C_A)$ as well as $OPT(A)$ are both integers, we can deduce from the (strong) inequality $\ell(C_A) < OPT(A)+d$ that $\ell(C_A) \le OPT(A)+(d-1)$. ■

C. Upper Bound on the Optimal Encoding

We now present a simple upper bound on the optimal encoding of any FIB table with an arbitrary number d of columns.

Consider a fixed-length encoding scheme I_A that encodes each column j using a fixed-length encoding (as seen in Example 2). Then in each column j, it uses codewords of size $W_j = \lceil\log_2 n_j\rceil$, since it needs at least $n_j > 2^{W_j-1}$ codewords to represent the n_j different values. As a consequence, the total width of each row in also fixed, and we have

$$\ell(I_A) = \sum_{j=1}^{d} W_j = \sum_{j=1}^{d}\lceil\log_2 n_j\rceil. \tag{8}$$

The fixed-length encoding yields an upper bound on the optimal encoding width of the FIB table.

Property 2. *Let A be a FIB table of d columns such that the set of distinct elements in the j^{th} column is S_j with $|S_j| = n_j \le 2^{W_j}$. Then,*

$$OPT(A) \le \sum_{j=1}^{d} W_j. \tag{9}$$

IV. Two-Column FIB Tables

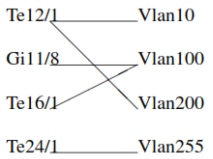
A popular class of FIB tables are L2 MAC tables that contain two columns. The first describes the Target Port, while the second can be viewed as an aggregation of columns representing a collection of other attributes. Indeed, we see in real-life table traces that these additional attributes are hardly ever used. Thus their aggregation has a relatively small set of possible values.

Therefore, from now on, we consider the special case of two-column FIB tables, i.e. FIB tables that satisfy $d=2$. We would like to fundamentally study this case to obtain some intuition on the problem. We show that in this case, in order to find its optimal encoding scheme, a FIB table A can be simply represented as a bipartite graph. We also suggest an analysis of the FIB encoding width of such two-column FIB tables.

For the sake of simplicity, we assume, unless mentioned otherwise, that the number of distinct elements in each of the

Target Port	Interface
Te12/1	Vlan10
Gi11/8	Vlan100
Te12/1	Vlan200
Te16/1	Vlan100
Te24/1	Vlan255
Te16/1	Vlan100

(a)



(b)

Fig. 2. A two-column forwarding table with $h = 6$ entries and $n = 2^W = 4$ (for $W = 2$) distinct elements in each column with (a) the table, and (b) its corresponding bipartite graph. The number of edges in the graph equals the number of distinct entries in the forwarding table.

$d = 2$ columns of A is the same, and that it is a power of two, i.e. $|S_1| = |S_2| = n = 2^W$.

A. The Optimal Encoding Width of a Two-Column FIB Table

The next theorem suggests a lower bound and an upper bound on the FIB encoding width of $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$ for $W \geq 2$.

Theorem 2. *Let A be a two-column FIB table with $|S_1| = |S_2| = n = 2^W$ for $W \geq 2$. Then, the encoding width of A satisfies*

$$W + 2 \leq OPT(A) \leq 2W. \tag{10}$$

Proof Outline: In [18], we present the full proof and also prove the tightness of these bounds by exposing, for each of them, a FIB table that achieves the bound.

The upper bound of $W + W = 2W$ can be simply achieved using Property 2, based on a fixed-length encoding. To show the lower bound, we consider two options for the codeword lengths of the elements in S_1. If all of them equal W bits, at least one of them shares a row with an element in S_2 encoded with two or more bits. Likewise, if the codeword lengths are not fixed, we have an element in S_1 with a codeword length of at least $W+1$ that shares a row with another arbitrary element in S_2 encoded in at least a single bit. Both cases then suffice to conclude. ∎

Any two-column FIB table $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$ can be represented by a corresponding bipartite graph G_A as follows. The two disjoint sets are the sets of distinct elements in each of the two columns. Thus, if an element appears in both of the two columns, it is represented by two different vertices in each of the two disjoint sets. Edges connect elements in the two sets if they appear at least once on the same row of the FIB table. Formally, we define the graph $G_A = <L + R, E>$ such that $L = S_1$, $R = S_2$ and $E = \{(x, y) | (\exists i \in [1, h]), (a_1^i, a_2^i) = (x, y)\}$. Therefore, duplicated rows have no influence on the construction of the bipartite graph, and the graph does not contain parallel edges. It is also easy to see the independence in the order of the rows of the FIB table.

Example 3. *Figure 2(a) presents an example of a two-column forwarding table with $h = 6$ entries and $n = 2^W = 4$ (for $W = 2$) distinct elements in both columns. The corresponding bipartite graph appears in Figure 2(b). The vertices on the left side of the graph represent the $n = 4$ distinct elements in the first column, while the vertices on the right side represent the n distinct elements in the second column. The number of edges in the graph equals the number of distinct entries in the forwarding table.*

Given a FIB encoding scheme $C_A = (\sigma_1, \ldots, \sigma_d)$ of $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$, we can present its FIB encoding width based on G_A as

$$\ell(C_A) = \max_{(x,y) \in E} \ell(\sigma_1(x)) + \ell(\sigma_2(y)). \tag{11}$$

From the construction of G_A, we can clearly see that the last equation is compatible with Definition 4.

The representation of a FIB table as a bipartite graph can help us to further understand the FIB encoding width based on tools from graph theory. The next theorem relates the existence of an independent set of a specific size in the bipartite graph G_A to the value of $OPT(A)$.

Theorem 3. *Let A be a two-column FIB table with $|S_1| = |S_2| = n = 2^W$ and let $G_A = <L+R, E>$ be its corresponding bipartite graph. If there does not exist an independent set of vertices $U = U_1 \bigcup U_2$ in G_A, so that $U_1 \subseteq L = S_1$, $U_2 \subseteq L = S_2$ and $|U_1| = |U_2| = \frac{n}{2} + 1$, then the optimal encoding width of A necessarily satisfies*

$$OPT(A) = 2W. \tag{12}$$

Namely, the optimal encoding width is achieved by the fixed-length encoding and it cannot be improved by any variable-length encoding.

Proof Outline: In any encoding scheme, there are at least $\frac{n}{2}+1$ elements in S_1 (respectively S_2) whose codeword lengths in σ_1 (σ_2) are at least W bits. If $OPT(A) < 2W$ then any two such elements cannot share an entry in A. ∎

B. Optimal Conditional Encoding of the Second Column

We now consider the conditional problem of finding an optimal two-column encoding *given that the encoding of the first column is known*.

Formally, given a two-column FIB table A and a known prefix encoding of one of its columns $\sigma_1 := \bar{\sigma}_1$, we want to find a prefix encoding σ_2 of the second column such that the encoding width $\ell(C_A)$ of the FIB encoding scheme $C_A = (\sigma_1 := \bar{\sigma}_1, \sigma_2)$ is minimized. We denote the FIB encoding width of such scheme by $OPT(A|\sigma_1 := \bar{\sigma}_1)$, i.e.

$$OPT(A|\sigma_1 := \bar{\sigma}_1) = \min_{C_A = (\sigma_1 := \bar{\sigma}_1, \sigma_2)} \ell(C_A). \tag{13}$$

We would like to suggest an algorithm to find such an optimal conditional encoding. Let A be a two-column FIB table with two sets of distinct elements in each of the columns S_1, S_2 and an encoding of the first column $\sigma_1 := \bar{\sigma}_1$. For $y \in S_2$, we denote by $\phi_{\bar{\sigma}_1}(y)$ the maximal codeword length $\ell(\bar{\sigma}_1(x))$ of an element $x \in S_1$ that shares a row with y in A.

$$\phi_{\bar{\sigma}_1}(y) = \max\{\ell(\bar{\sigma}_1(x)) | x \in S_1, (x, y) \in E\}. \tag{14}$$

(a) (b)

Fig. 3. Two examples of bipartite graphs with $n = 2^W = 8$ (for $W = 3$) vertices in each graph side representing two two-column FIB tables A_1 and A_2. In the first graph (a), there is a perfect match and the size of the (minimal) edge cover is $8 = n = \alpha \cdot n$ with $\alpha = 1$ and $OPT(A_1) = 2W = 6$. In the second (b), the size of the edge cover is $14 = \alpha \cdot n$ with $\alpha = 1.75$ and $OPT(A_2) \geq 5$.

We obtain the following closed-form solution:

Theorem 4. *The optimal conditional FIB encoding width of A satisfies*

$$OPT(A|\sigma_1 := \bar{\sigma}_1) = \left\lceil \log_2 \Big(\sum_{y \in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil . \tag{15}$$

Proof Outline: We first prove that there exists a prefix encoding $\bar{\sigma}_2$ of S_2 such that $(\bar{\sigma}_1, \bar{\sigma}_2)$ achieves the minimal encoding width within all the FIB encoding schemes $(\bar{\sigma}_1, \sigma_2)$ and satisfies $(\forall y \in S_2), \phi_{\bar{\sigma}_1}(y) + \ell(\bar{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1)$. Then, we use this formula, Kraft's inequality, and convexity arguments to conclude. We also further explain in [18] how to obtain such a solution. ∎

V. Lower Bounds on the Optimal Encoding Width of Two-Column FIB Tables

We would like now to suggest an additional lower bound on the optimal encoding width of a given two-column FIB table A as defined above. We can calculate this bound without solving the relaxed FIB encoding problem.

Let again $G_A = <L + R, E> = <S_1 + S_2, E>$ with $|S_1| = |S_2| = n = 2^W$. In our case, note that G_A does not include any isolated vertices, i.e. any vertex is connected to at least one vertex and the value it represents appears in at least one of the rows of A. Therefore, an edge cover always exists. For a graph G, we denote by $\rho(G)$ the edge covering number of G, i.e. the minimal number of edges in an edge cover of G. Let $\alpha = \rho(G_A)/n$. The next theorem suggests a lower bound on the optimal encoding width of A based on the value of $\rho(G_A)$.

Theorem 5. *The optimal encoding width of A satisfies*

$$OPT(A) \geq \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil . \tag{16}$$

Proof Outline: The full proof can be found in [18]. We first prove that the edge covering number $\rho(G_A)$ of G_A satisfies

$$n \leq \rho(G_A) \leq 2 \cdot (n - 1). \tag{17}$$

Then, we consider a subset of the rows of A of size $\rho(G_A) = \alpha \cdot n$, such that the corresponding edges in G_A belong to a minimal edge cover, and use it to conclude. ∎

Example 4. *Figure 3 illustrates the two corresponding bipartite graphs for two-column FIB tables A_1 and A_2, with $n = 2^W = 8$ (for $W = 3$) distinct elements in each column.*

For A_1, the bipartite graph includes a perfect match and the size of the edge cover is $8 = n = \alpha \cdot n$ with $\alpha = 1$. Theorem 5 implies that $OPT(A_1) \geq \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil = 2W$. By the upper bound from Theorem 2 we deduce that $OPT(A_1) = 2W$.

For A_2, the size of the minimal edge cover is $14 = 2 \cdot (n - 1) = \alpha \cdot n$ with $\alpha = 1.75$. Thus $OPT(A_2) \geq \left\lceil \frac{2 \cdot (3 + 1.75 - 1)}{1.75} \right\rceil = 5$. Further, by encoding the first element in a column in a single bit and others in four bits, we have that indeed $OPT(A_2) = 5$.

VI. Supporting Updates

Forwarding tables need to support updates. These updates can include an insertion of a new entry, a deletion of an existing entry, or a modification of some of the fields of an entry. In this section, we discuss how such updates can be supported.

Let P be the fixed allocated number of bits for the encoding of each entry. If the support of such updates is required, we assume that each entry includes an additional bit that indicates its validity.

In general, the update process includes two steps: *an immediate step required for coherency, and another procedure that can occur offline to improve performance*. This second step can be run, for instance, either periodically in time or after a fixed number of updates. Meanwhile, in addition to the encoded table, we make use of a small dedicated memory for keeping a small number of entries.

Dealing with a change in an entry or with an insertion can be done in a similar way. We describe several possible scenarios. First, *if all the elements in the updated entry appear in the current dictionaries*, we simply encode a new entry based on these dictionaries. If its total width is not greater than P, no further changes are required. If it is greater than P, we set the original entry (in case of a change) as invalid and temporarily add the new entry (without encoding it) to the small memory.

If the updated entry includes a new element, we try to associate it with a codeword. In a specific column, we can do so only if the sum that appears in the Kraft's inequality is smaller than one. Further, the minimal possible codeword length depends on the value of this sum. For the sake of simplicity, we suggest to allocate for each new element a codeword with the minimal possible length. If we cannot allocate any new codeword in one of these columns (due to an equality in the corresponding Kraft's inequality), or if the allocated codewords yield an entry that is too long, the updated entry is again added to the small memory.

Dealing with *an entry deletion* is easy and requires setting the entry as invalid. Reduction in the maximal width of the table after an entry removal may be achieved by running the compression algorithm as part of the offline procedure.

The offline process includes the calculation of an efficient encoding for the existing encoded entries and for the unencoded entries stored in the small memory. Later, this memory is cleared. The value of P may be changed, as well as the encoded

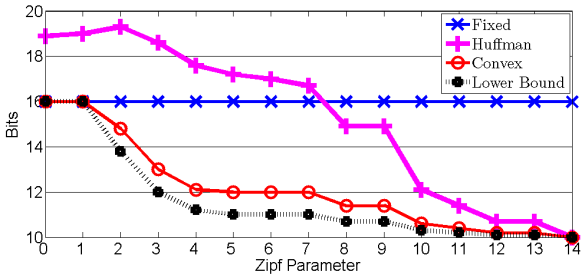
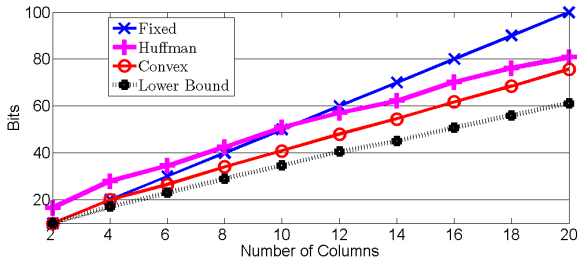
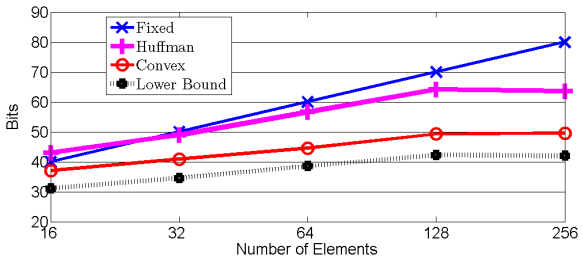


Fig. 4. Effectiveness of the encoding algorithm on synthetic FIB tables with $d = 2$ columns and $W = 8$, as a function of the Zipf parameter. Elements are selected according to the Zipf distribution with the given parameter. By Theorem 2, the optimal encoding width is at least $W + 2 = 10$ and at most $2W = 16$. Since $d = 2$, our suggested *convex* scheme takes at most $d-1 = 1$ bit more than the theoretical lower bound. In comparison, the fixed-length encoding has a fixed width of $2W = 16$ bits, while the Huffman encoding sometimes achieves a greater width.



(a) Effectiveness of the encoding algorithm as a function of the number of columns with $2^W = 32$ distinct elements per column.



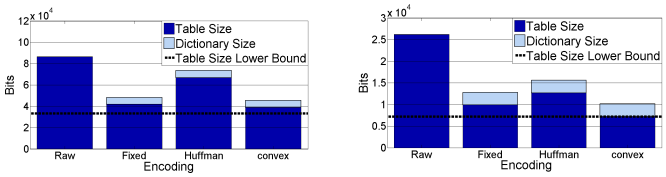
(b) Effectiveness of the encoding algorithm as a function of the number of distinct elements per column with $d = 10$ columns.

Fig. 5. Analysis of the encoding algorithm as a function of table parameters.

version of existing entries. It is possible to endow the table encoding algorithm with better update capabilities by choosing an encoding with a slightly larger value of P that guarantees short codeword lengths for possible future inserted elements. Meanwhile, between two offline processes, any lookup should consider the encoded entries in the FIB and the unencoded entries in the small dedicated memory.

VII. Experimental Results

We now turn to conducting experiments to examine the distribution of the optimal encoding width and the performance of our suggested encoding algorithm from Section III-B. In the experiments, we rely on both synthetic and real-life FIB tables.



(a) United States (b) China

Fig. 6. Total memory size of encoded real-life tables including the dictionary size.

A. Effectiveness on Synthetic FIB tables

We first evaluate our suggested scheme on *synthetic two-column FIB tables*. Each table has 1000 entries with $2^W = 256$ (for $W = 8$) distinct elements in each of the $d = 2$ columns. To generate the table, we first make sure that each element appears at least once in each column by adding an entry containing it with an additional random element in the other column. This additional random element is selected according to a Zipf distribution of parameter s. Then, we add additional entries with $d = 2$ random elements selected according to the same distribution. Intuitively, a Zipf distribution with a lower parameter s is closer to the uniform distribution.

First, in Fig. 4, we compare our suggested scheme, denoted "convex", with the fixed-length and Huffman encoding schemes. The results are based on the average of 10 simulations. We present the lower bound given by the fractional solution of the relaxed optimization problem (Section III-B). We also show our suggested scheme, which takes the ceiling of the fractional solution of the relaxed problem, unless it exceeds $2W$, in which case it simply uses fixed-length encoding. As suggested by Theorem 1, our scheme is always within $d-1 = 1$ bit of the fractional-optimum lower bound. For instance, for a Zipf parameter $s = 4$, the widths for the Huffman, fixed-length, and our suggested encodings are respectively 17.6, $2W = 16$, and 12.1, while the lower bound is 11.2. More generally, the lower the Zipf parameter, the closer the lower bound to $2W$.

Next, in Fig. 5, we plot the encoding efficiency as a function of the number of columns (a) and of the number of elements appearing in each column (b). In both plots we take $h = 1000$ entries and a Zipf parameter $s = 2$. In Fig. 5(a) the number of elements is 32, i.e. $W = 5$, and in Fig. 5(b) the number of columns is $d = 10$. We can see that in (a), Huffman becomes better for the worst-case entry width. Intuitively, it is because the ratio of the standard deviation of the sum of the encodings of d columns to the expected sum decreases with d as $\frac{1}{\sqrt{d}}$.

B. Real-Life FIB tables

We also conduct experiments on two typical real-life enterprise-network tables. The tables are collected from two different switches of two different switch vendors, each switch using a different number of columns. All tables were collected in enterprise networks, the first in the United States with 2790 entries and the other in China with 903 entries. For each table, we present the size of the original raw table without compression. We compare it to the total size of the compressed table,

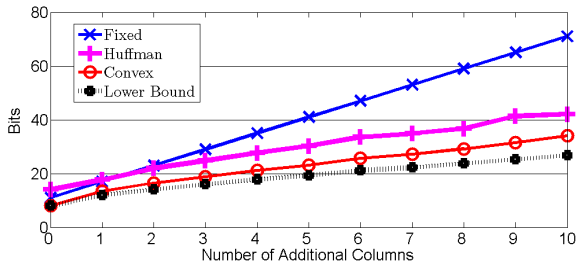


Fig. 7. Effectiveness of the encoding algorithm on the extended table (based on the China table), as a function of the number of columns.

including the dictionary for three compression algorithms: the fixed-length encoding, the Huffman encoding and our suggested encoding. The results are presented in Fig. 6.

For instance, in the second real-life table (China), presented in Fig. 6 (b), we consider the 903 entries of two columns representing the VLAN and the Target-Port fields. Without encoding, each entry is allocated 29 bits per entry and thus the raw data without encoding requires $903 \cdot 29 = 26187$ bits. Using the three encodings, the size of the dictionaries is almost fixed and equals approximately 2900 bits. An entry width of 11 and 14 bits is achieved in the fixed-length encoding and in the Huffman encoding, respectively, while in the proposed encoding we achieve an entry width of only 8 bits. This leads to an improvement of 20.7%, 34.9%, and 61.3% in the total memory size (including the dictionary) compared to the fixed-length, Huffman and Raw solutions, respectively.

C. Extrapolated Tables

We want to extrapolate the behaviors of these schemes from real-life limited data sets to the larger expected data sets to be encountered in future network architectures. The approach we take herein is to amplify the available data sets by scaling out the number of columns in the table, so as to model a table with a scaled-out number of attributes.

We extend the China table by duplicating and appending the port column several times. In each new column, we permute the appearances of the elements uniformly at random. The motivation is to model additional forwarding attributes that would follow a distribution that is similar to an existing one. Rearranging the row locations of the duplicated elements adds randomness. As seen in Fig. 7, for each additional column the fixed encoding increases by a constant, as expected since the number of unique elements in each column does not vary. The Huffman and convex encodings increase at a slower pace, thereby increasing the encoding effectiveness as the number of columns grows. Of course, since this table is extended synthetically, we urge caution in over-interpreting the results.

VIII. Conclusion

In this paper, we saw how datacenter virtualization is causing a dramatic rise in the number of entries in forwarding tables, such that it becomes impossible to implement forwarding tables in current memory sizes without any compression algorithm. We then investigated the compressibility of forwarding tables, by introducing a novel forwarding table architecture with separate encoding in each column. Our architecture supports fast random accesses and fixed-width memory words. We also later explained how our architecture can handle table updates.

Later, we suggested an encoding whose per-entry memory requirement is guaranteed to be within a small additive constant of the optimum. Finally, we evaluated this new scheme against the fixed-width and Huffman schemes, using both synthetic and real-life forwarding tables from different switches, switch vendors, and network country locations. As future work, we plan to check how using additional hardware such as CAM and TCAM can help in further compressing the forwarding tables.

IX. Acknowledgment

We would like to thank Yishay Mansour and Haim Kaplan for their helpful participation and suggestions.

This work was partly supported by the European Research Council Starting Grant No. 210389, by a European Commission Marie Curie CIG grant, by the Israel Science Foundation grant No. 1241/12, by the German-Israeli Foundation for Scientific Research and Development, by the Intel ICRI-CI Center, by the Hasso Plattner Center for Scalable Computing and by the Israel Ministry of Science and Technology. Ori Rottenstreich is the Google Europe Fellow in Computer Networking, an Andrew and Erna Finci Viterbi Fellow and a Jacobs-Qualcomm Fellow.

References

[1] I. Gashinsky, "Datacenter scalability panel," in *NANOG 52*, 2011.
[2] R. N. Mysore *et al.*, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM*, 2009.
[3] T. Narten, M. Karir, and I. Foo, "Problem statement for ARMD," IETF, draft-ietf-armd-problem-statement, work in progress, 2012.
[4] G. Hankins, "Pushing the limits, a perspective on router architecture challenges," in *NANOG 53*, 2011.
[5] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," IETF, RFC 4984, 2007.
[6] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley, 2006.
[7] T. Narten, "On the scalability of internet routing," IETF, draft-narten-radir-problem-statement, work in progress, 2010.
[8] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the aggregatability of router forwarding tables," in *IEEE Infocom*, 2010.
[9] R. Draves, C. King, V. Srinivasan, and B. Zill, "Constructing optimal IP routing tables," in *IEEE Infocom*, 1999.
[10] Q. Li, D. Wang, M. Xu, and J. Yang, "On the scalability of router forwarding tables: Nexthop-selectable FIB aggregation," in *IEEE Infocom Mini-Conference*, 2011.
[11] P. Francis *et al.*, "FIB suppression with virtual aggregation," IETF, draft-ietf-grow-va, work in progress, 2011.
[12] M. Pöss and D. Potapov, "Data compression in Oracle," in *VLDB*, 2003.
[13] R. Johnson, V. Raman, R. Sidle, and G. Swart, "Row-wise parallel predicate evaluation," *PVLDB*, 2008.
[14] K. Stolze, V. Raman, R. Sidle, and O. Draese, "Bringing BLINK closer to the full power of SQL," in *BTW*, 2009.
[15] Y. Kanizo, D. Hay, and I. Keslassy, "Optimal fast hashing," in *IEEE Infocom*, 2009.
[16] A. Kirsch and M. Mitzenmacher, "The power of one move: Hashing schemes for hardware," *IEEE/ACM Trans. Netw.*, 2010.
[17] Y. Kanizo, D. Hay, and I. Keslassy, "Hash tables with finite buckets are less resistant to deletions," *Computer Networks*, 2012.
[18] O. Rottenstreich *et al.*, "Compressing forwarding tables," Technion, Tech. Rep. TR12-03, 2012. [Online]. Available: http://webee.technion.ac.il/~isaac/papers.html
[19] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Society for Industrial Mathematics, 1987, vol. 13.