CCF Transactions on Networking
https://doi.org/10.1007/s42045-018-0010-1

REGULAR PAPER

Mathematical analysis on forwarding information base compression

Tong Yang[1,2] · Jinyang Li[1,2] · Chenxingyu Zhao[1,2] · Gaogang Xie[1,2] · Xiaoming Li[1,2]

Received: 29 June 2018 / Accepted: 22 November 2018

Abstract
With the fast development of Internet, the size of routing table in the backbone router continues to grow rapidly. forwarding information base (FIB), which is derived from routing table, is stored in line-card to conduct routing lookup. Since the line-card's memory is limited, it would be worthwhile to compress the FIB for consuming less storage. Therefore, various FIB compression algorithms have been proposed. However, there is no well-presented mathematical support for the feasibility of the FIB compression solution, nor any mathematical derivation to prove the correctness of these algorithms. To address these problems, we propose a universal mathematical method based on the *Group* theory. By defining a *Group* representing the longest prefix matching rule, the bound of the worst case of FIB compression solution can be figured out. Furthermore, in order to guarantee the *ultimate correctness* of FIB compression algorithms, routing table equation test is proposed and implemented to verify the equivalence of the two routing tables before and after compression by traversing the 32-bit IP address space.

Keywords FIB compression · Group · Trie-transformation · LPM · RTET

1 Introduction

The backbone routing table has been growing at an exponential rate, driven mainly by multi-homing and the rapid development of mobile communication (Meng et al. 2005). The fast increasing routing table incurs fast increasing FIB. The line-card that holds the FIB usually adopts fast memory, which is expensive and difficult to scale, and it would be worthwhile to improve the memory efficiency by compressing the FIB. Besides, for the routing lookup schemes based on software (Waldvogel et al. 1997; Degermaerk et al. 1997; Nilsson et al. 1998), FIB compression can be used to reduce their memory requirements; for the routing lookup algorithms based on TCAM (Zheng et al. 2006; Lin et al. 2007; Yang et al. 2012), FIB compression can be used to reduce the hardware cost and power consumption. Furthermore, FIB compression is applicable to any longest matching prefix (LPM) database. Therefore, a variety of FIB compression algorithms are proposed (Draves et al. 1999; Cain 2002; Zhao et al. 2010; Li et al. 2011, 2013; Liu et al. 2010; Yu 2010). These algorithms compress the routing table by transforming the binary trie[1] structure.

In addition, as is known to all, routing table lookup adopts the longest prefix matching (LPM) rule because of the introduction of classless inter-domain routing (CIDR). Due to CIDR, the routing tables' prefixes are overlapped, which means that some prefixes are a part of others. This brings many negative effects on the performance of routing lookup and incremental update (Yang et al. 2012). There are mainly two overlap elimination algorithms: Leaf-pushing (Srinivasan and Varghese 1999) and ONRTC (Yang et al.

Co-primary authors: Tong Yang and Jinyang Li.

✉ Jinyang Li
lijinyang@pku.edu.cn

Tong Yang
yangtongemail@gmail.com

Chenxingyu Zhao
dkzcxy@pku.edu.cn

Gaogang Xie
xie@ict.ac.cn

Xiaoming Li
lxm@pku.edu.cn

[1] Peking University, Beijing, China

[2] Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, China

[1] Trie is a tree-based data structure allowing the organization of prefixes on a digital basis by using the bits of prefixes to direct the branching (Ruiz-Sánchez et al. 2001).

Published online: 07 December 2018

2012) algorithm. They can totally eliminate the overlap also by transforming the binary trie.[2]

However, is FIB compression solution feasible? What's the worst case of the FIB compression solution? How to guarantee the correctness of trie-transformation algorithms? Current FIB compression algorithms just compress the routing table, regardless of the size and structure of the routing table. In contrast, the feasibility, effectiveness and correctness of FIB compression algorithms are emphasized and well-studied in this paper.

(a) *Feasibility and effectiveness* According to the information theory, it is definite that the compressed routing table holds the information equivalent to the original one. Therefore, if and only if there is redundancy in the original routing table, the FIB compression solution is feasible. Then is there redundancy in the routing table? What's the premise of the existence of redundancy? After a profound data mining of the routing tables, we find that although the routing table is rapidly growing (some backbone routers have more than 400 K FIB entries today), the port number of a router is extremely limited (ranging from 3 to 80) and almost static. This observation intuitionally gives a positive answer to the existence of redundancy. Fortunately, the redundancy caused by the almighty gap between the prefix number and port number in the routing table can be quantized by Pigeonhole Principle. Based on this observation, we also deduce the bound of the worst case of the FIB compression solution in this paper.

(b) *Correctness* After an in-depth study, we reveal that the LPM rule can be well expressed by the regular expression syntax. We also find that the LPM rule can be well expressed by the *Group*[3] theory. Based on these two advancements, two basic equivalent atomic models are induced—election model and representative model. We insist that all the trie-transformation algorithms can be proven by these two fundamental atomic models.

Actually, FIB compression algorithm is a tough and error-prone task during the algorithm design and implementation. In order to guarantee the *ultimate correctness* of FIB compression algorithms, we propose routing table equation test (RTET) to verify the equivalence of the two routing tables before and after compression by traversing the 32-bit IP address space.

Specifically, the main contributions of this paper lie in the following aspects.

- We propose a universal mathematical method based on a new defined *Group*, and apply this method to four classical FIB compression algorithms.
- We compute the bound of the worst case of various FIB compression solutions.
- We propose and implement routing table equation test (RTET) for the first time, to verify the equivalence of the two tries before and after binary trie transformation by traversing the 32-bit IP address space. At the end, we implement and verify four classical algorithms by RTET.

The remaining parts of the paper are organized as follows. Section 2 surveys the related work. Section 3 elaborates on a novel mathematical method which can prove the correctness of trie-transformation algorithms. The bound of the worst case of FIB compression solution is analyzed in Sect. 4. Section 5 applies this mathematical proof to four classical FIB compression algorithms. The *ultimate correctness* of FIB compression algorithm is guaranteed by RTET, which is illustrated in Sect. 6. Finally, we conclude this paper in Sect. 7.

2 Related work

IRTF RRG (2014) and IETF GROUP (2015) have been working on the issues about routing scalability for years. Forwarding Information Base[4] (FIB) compression is a local solution and needs no change to the existing routing protocols, and the representative papers are (Draves et al. 1999; Cain 2002; Zhao et al. 2010; Li et al. 2011, 2013).

Richard Draves et al. (1999) proposed the famous ORTC algorithm, which constructs optimal routing tables. However, ORTC algorithm has not been applied to real routers, for the reason that it is so complicated that it consumes a lot of time and memory, which is not conducive to incremental updates. The core operations of ORTC are 'UNION' and 'AND', and thus the correctness of these two operations is proven in this paper.

In Cain (2002), a patent technology proposed a compression algorithm, which is simple and fast, but its compression ratio is not high. There is no mathematical proof in this patent. For convenience, it is called patent algorithm in this paper.

Xin Zhao et al. (2010) proposed a 4-level algorithm. The first two levels are plain, and cannot achieve a high

[2] Both FIB compression and overlap elimination algorithms transform the binary trie, thus they are called trie-transformation algorithms in this paper.

[3] Group (mathematics) (Vvedensky 2005) is a set together with a binary operation satisfying certain algebraic conditions.

[4] FIB is also known as forwarding table, which is stored in line-cards to forward data packets. Each entry of FIB stores a prefix and the corresponding next-hop, such as 200.45.65.0/24:40. It suggests that if an incoming IP address matched the prefix 200.45.65.0/24 by LPM, this packet should be forwarded to the interface 40 (40 is also related to a corresponding next-hop IP address).

compression ratio. Although the third and fourth levels can achieve high compression ratio, they can only deal with non-routable space cases. The non-routable packets mean those that should be dropped, because no next-hop could be found, but they are forwarded anyway (weak correctness). It is called roaming garbage in this paper. Our real traffic trace shows that the amount of traffic caused by roaming garbage could be up to 0.31% of the total traffic and it covers 0.38% of the whole IP address space. There is no mathematical proof in that paper, either. The 4-level algorithm consists of four transformation models. The first two models with strong correctness as well as the other two with weak correctness are proved to be right in this paper.

Li et al. (2011, 2013) proposed the NSFIB compression algorithm, which can achieve a much better compression ratio than ORTC and 4-level by taking advantage of multiple next hops, the overhead is sometimes choosing the suboptimal routing path. The author also presented several practical choices to build the sets of alternative next hops for the prefixes, and developed an optimal online algorithm with constant running time. NSFIB algorithm is potentially applied to multicast network (Li et al. 2011).

As a forementioned, there are mainly two algorithms to eliminate overlap: leaf-pushing (Srinivasan and Varghese 1999) and ONRTC (Yang et al. 2012). In Srinivasan and Varghese (1999), leaf-pushing algorithm is proposed to eliminate overlap. This algorithm is simple, which just pushes the internal nodes to leaf nodes, inevitably causing the expansion of routing table size. The correctness of this algorithm isn't proven theoretically, either. In order to reduce the routing table size of leaf-pushing, ONRTC algorithm in Yang et al. (2012) is proposed to construct optimal non-overlap routing tables, achieving 71% compression ratio according to the experimental results.

There is another kind of FIB compression algorithm, and we call it entropy compression in this paper. Rétvári et al. (2013) and IRTF Routing Research Group (2014 applied the theory of information entropy to IP prefixes for the first time, and there are two successors (Rottenstreich et al. 2013; Korosi et al. 2014). This kind of compression algorithm transforms the FIB into new format other than prefixes, and thus can hardly work with TCAM or SRAM pipeline. The goal of this kind of algorithm is to approach the theory bound of information entropy. The overhead of entropy compression is difficult to perform incremental update.

All these trie-transformation algorithms do not emphasize the mathematical analysis and proof. This motivates our work reported in this paper.

3 Mathematic proof

In this section, a novel mathematical method, which can be used to prove the correctness of trie-transformation algorithms, is proposed. Firstly, a new *Group* is defined, and its four conditions are proven. Secondly, two fundamental atomic models are proven to be right based on this new defined *Group*. We insist that all the trie-transformation algorithms can be proven by these two atomic models, and their applications are highlighted in Sect. 5. We insist that for all FIB compression algorithms, only those compressing the routing table by transforming the binary trie structure can use this proposed theory, and others cannot.

3.1 Group definition

Prefixes are a series of bits. They can be well represented by regular expression syntax (http://www.regular-expressions.info/reference.html), and the symbols frequently used in this paper are defined below.

- A is a node in the trie, while (A) represents node A's prefix. Solid nodes have next-hop, while hollow nodes have not.
- (AB) represents the bit string of the path between node A and B, while no solid nodes appear in the path.
- If A is an ancestor of B, then $A \subset B$.
- L(A) represents the prefix length of node A.
- P represents a trie, and (A) represents a prefix, then P(A) means the next-hop of prefix (A) in trie P.
- $(\widetilde{A})$ represents a prefix with the same length of (A), but it is different from (A). $P(\widetilde{A})$ means the next hop of prefix $(\widetilde{A})$ in trie P.
- (A*) is a 32-bit prefix, and (A) is a part of (A*). P(A*) means the next hop of prefix (A*).

In mathematics, a *Group* (Vvedensky 2005) is a set of elements together with a binary composition law, which must satisfy four conditions: closure, associativity, identity, and invertibility.

Definition 1 LPM Group.

Let G be the LPM *Group,* and G = Z. The operation on LPM *Group* is XOR, which is illustrated as follows:

$$\forall x, y \in G$$

$$\mathbf{x} \oplus \mathbf{y} = \begin{cases} \mathbf{x} + \mathbf{y}, & \mathbf{xy}(\mathbf{x} + \mathbf{y}) = \mathbf{0} \\ \mathbf{y}, & \mathbf{x} > 0, y > 0 \\ \textbf{meaningless}, & \textbf{other else} \end{cases} .$$

As shown in Fig. 1, the function $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$ is plotted in three-dimensional space.

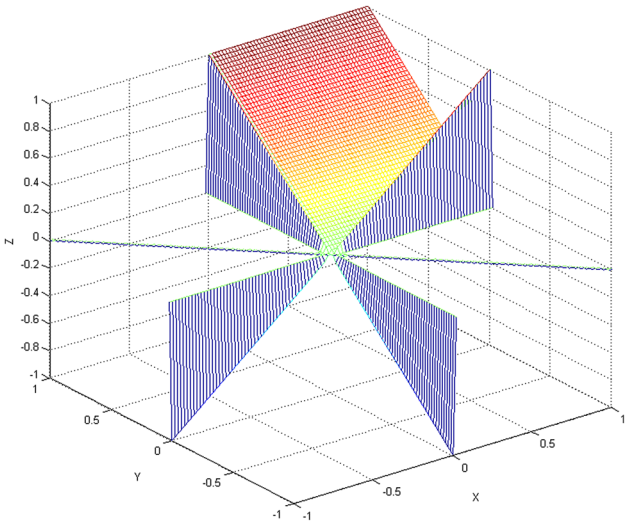


Fig. 1 LPM group in three-dimensional space

Condition 1 Closure

Proof

$\forall x, y \in G$, obviously, $x \oplus y \in G$.

Therefore, LPM Group satisfies *Closure*. □

Condition 2 Associativity

Proof

$\forall$x, y, z $\in$ G

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$.

(1) If $x = 0, (x \oplus y) \oplus z = y \oplus z, x \oplus (y \oplus z) = y \oplus z$.
Therefore,

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$.

Similarly, if y = 0 or z = 0, $(x \oplus y) \oplus z = x \oplus (y \oplus z)$.

(2) x $\neq$ 0 and y $\neq$ 0 and z $\neq$ 0..

(2.1) If $x + y = 0$, in order to make $(x \oplus y) \oplus z$ and $x \oplus (y \oplus z)$ meaningful, $y + z$ must be zero. Therefore,

$(x \oplus y) \oplus z = 0 \oplus z = z$,

$x \oplus (y \oplus z) = x \oplus 0 = x = z$,

$\therefore (x \oplus y) \oplus z = x \oplus (y \oplus z)$.

(2.2) If $x > 0$, *and* $y > 0$

$(x \oplus y) \oplus z = y \oplus z = z$,

$x \oplus (y \oplus z) = x \oplus z = z$.

$\therefore (x \oplus y) \oplus z = x \oplus (y \oplus z)$.

Therefore, LPM **Group** satisfies *Associativity*.

□

Condition 3 Identity

Proof

$$0 \oplus y = \left\{ \begin{array}{l} 0, \ y = 0 \\ y, \ y > 0 \end{array} \right. \Rightarrow 0 \oplus y = y$$

$$y \oplus 0 = \left\{ \begin{array}{l} 0, \ y = 0 \\ y, \ y > 0 \end{array} \right. \Rightarrow y \oplus 0 = y \Bigg\} \Rightarrow 0 \text{ is the identity.}$$

Therefore, LPM **Group** satisfies *Identity*. □

Condition 4 Invertibility

Proof

$\forall x \in G, x \oplus (-x) = (-x) \oplus x = 0$.

Therefore, − x is the inverse of x.

According to the above four conditions, it can be concluded that G is a **Group**.

*LPM **Group*** is used to describe the matching process and results of prefixes in this paper, and thus we define the next-hop and induce Theorem 1 in the following. □

Definition 2 *P(R)*

$\forall$IP address R, R = [0,1]{32}, the match result of each bit is S_i for IPv4, i = 1, 2, … , 32; for IPv6, i = 1, 2, … , 128; According to the longest prefix matching rule, the next-hop of R is $P(R) = S_1 \oplus S_2 \oplus S_3 \oplus \ldots S_{32} = \oplus_{i=1}^{32} S_i$.

Theorem 1 *If the match results of every section of two prefixes are same, then the next-hops of the two prefixes are same.*

Proof

$$P1(R) = \oplus_{i=1}^{32} S_i = (\oplus_{i=1}^{t1} S_i) \oplus (\oplus_{i=t1+1}^{t2} S_i) \oplus (\oplus_{i=t2+1}^{t3} S_i) \oplus \ldots \oplus (\oplus_{i=tn+1}^{32} S_i)$$
$$P2(R) = \oplus_{i=1}^{32} V_i = (\oplus_{i=1}^{t1} V_i) \oplus (\oplus_{i=t1+1}^{t2} V_i) \oplus (\oplus_{i=t2+1}^{t3} V_i) \oplus \ldots \oplus (\oplus_{i=tm+1}^{32} V_i).$$

Suppose $P1_k = \oplus_{i=tx+1}^{k} S_i, P2_k = \oplus_{i=tx+1}^{k} V_i$, then

$P1(R) = P1_{t1} \oplus P1_{t2} \oplus \ldots \oplus P1_{32}$

$P2(R) = P2_{t1} \oplus P2_{t2} \oplus \ldots \oplus P2_{32}$

$P1_k = P2_k, k = t1, t2, \ldots, 32.$

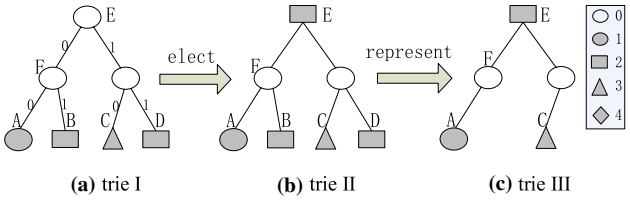
Therefore, $P1(R) = P2(R)$.

This theorem can be used to prove the equivalence of the next-hop of two tries section by section with regard to one IP address. □

Theorem 2 *Decision Theorem*

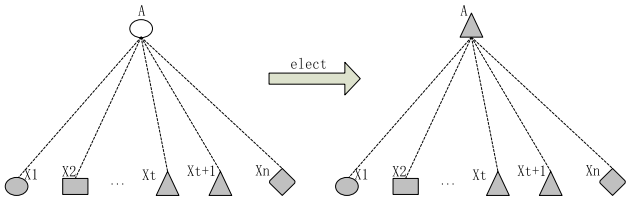
The necessary and sufficient condition that two tries are equivalent is that the next-hops are equal in the two tries for any IP addresses by LPM rule.

Obviously, this *Decision Theorem* naturally holds. Combining Theorem 1 and Theorem 2, we can prove the equivalence of two tries (or two models) section by



(a) trie I **(b)** trie II **(c)** trie III

Fig. 2 An example of election and representative model



(a) The original trie **(b)** Trie after election

Fig. 3 Election model

section. Above all, two basic models which are used to prove other models in this paper are illustrated below.

3.2 An example of election and representative model

As shown in Fig. 2, each solid node represents a prefix, owning a next-hop, while the hollow nodes suggest that there is no prefix in this node, and we say their next-hops are 0. The next-hop is represented by the shape of the solid node. For convenience, four next-hops are introduced: solid ellipse, solid rectangle, solid triangle, and solid diamond, representing the next-hop of 1, 2, 3, and 4, respectively. For example, ▭ indicates its next-hop is 2, while ◯ implies its next-hop is 0.

In essence, all the trie-transformation algorithms follow a process that is similar to the election process of the democratic society. Each node has a next-hop, while each candidate has a vote (Dai et al. 2016a, b, 2018; Zhu et al. 2017). Actually, any candidate's next-hop can be selected as representative, resulting in different compression ratios. All the nodes which own the same next-hop with the representative can be deleted.

To make a clearer picture of election and representative model, an intuitive example is given in Fig. 2. Figure 2a is the original trie, Fig. 2b is the trie after election, and Fig. 2c is the trie after representative.

Election Nodes A, B, C, and D are four 'candidates' nodes, participating in 'election'. Obviously, hop 2 (rectangle B and D) should be elected as 'representative', and thus the next-hop of node E is set to 2.

Representative Node E executes its right of representative: keeping its voters silent, i.e., deleting its voters (node B and D).

3.3 Election and representative models

We insist that all the trie-transformation algorithms can be proven by two basic transformation models: election model and representative model.

3.3.1 Election model

Election model Two or more nodes elect their common ancestor node, and no solid node appears in the path from the candidate nodes to the common ancestor node. Any candidates can be elected as a representative, resulting in different compression ratios.

Election models can work on both binary trie and multi-bit trie. As shown in Fig. 3, the next-hop of Node X_i is N_i, the count of N_i is C_i. These candidate nodes must satisfy the following constraints.

- Node X_i is the elected representative, and it can own subtree.
- $A \subset X_i,\ i = 1, 2, 3, \ldots, n.$
- There is no solid node in the path between Xi and A.
- There is no missing node.

Election result If such t exists: $\forall j! = t,\ C_j \leq C_t$ holds, then X_t is the elected representative. If such t does not exist, election fails. Then the common ancestor's next-hop is set to NULL, and participates in the next round election. In this way, an optimal compression ratio can be achieved.

Proof ∀IP address R, obviously, $L(R) = K, R = [0, n]\{K\}$. Suppose $R = [0, n]\{L(A)\}[0, n][0, n]\{K - L(A) - 1\}$.

Step 1 Match [0, 1] $\{L(A)\}$

$$
\left.
\begin{aligned}
[0,1]\{L(A)\} = (A) \Rightarrow & \begin{cases} P1([0,1]\{L(A)\}) = P1(A) \\ P2([0,1]\{L(A)\}) = P2(A) \end{cases} \\
[0,1]\{L(A)\} \neq (A) \Rightarrow & \begin{cases} P1([0,1]\{L(A)\}) = P1(\widetilde{A}) \\ P2([0,1]\{L(A)\}) = P2(\widetilde{A}) \end{cases} \\
& P1(A) = P2(A) \\
& P1(\widetilde{A}) = P2(\widetilde{A})
\end{aligned}
\right\}
\Rightarrow \begin{cases} P1([0,1]\{L(A)\}) = P1_{s1} \\ P2([0,1]\{L(A)\}) = P2_{s1} \end{cases}.
$$

Step 2 Match $[0,1]$

$$
[0,1] = i, \quad i = 1,2,3,\ldots, \quad n \Rightarrow \left.
\begin{aligned}
& \begin{cases} P1([0,1]) = P1\left(X_i\right) \\ P2([0,1]) = P2\left(X_i\right) \end{cases} \\
& P1(X_i) = P2(X_i)
\end{aligned}
\right\}.
$$

$$
\Rightarrow P1([0,1]) = P2([0,1]) = P_{s2} \neq 0
$$

Step 3 Match $[0,1]\{K - L(A) - 1\}$

$$
\left.
\begin{aligned}
& [0,1] = i, \quad i = 1,2,\ldots,n \\
\Rightarrow & \begin{cases} P1([0,n]\{K - L(A) - 1\}) = P1\left(X_i *\right) \\ P2([0,n]\{K - L(A) - 1\}) = P2\left(X_i *\right) \end{cases} \\
& P1\left(X_i *\right) = P2\left(X_i *\right)
\end{aligned}
\right\}
$$

$$
\Rightarrow P1([0,n]\{K - L(A) - 1\}) = P2([0,n]\{K - L(A) - 1\}) = P_{s3}
$$

According to Steps 1, 2, and 3,

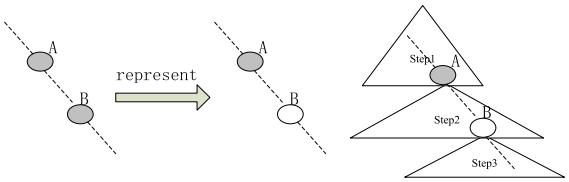
$$
\begin{aligned}
P1(R) &= P1([0,n]\{L(A)\}[0,n][0,n]\{K - L(A) - 1\}) \\
&= P1([0,n]\{L(A)\}) \oplus P1([0,n]) \oplus P1([0,n]\{K - L(A) - 1\}) \\
&= P1_{s1} \oplus P_{s2} \oplus P_{s3}, \\
P2(R) &= P2([0,n]\{L(A)\}[0,n][0,n]\{K - L(A) - 1\}) \\
&= P2([0,n]\{L(A)\}) \oplus P2([0,n]) \oplus P2([0,n]\{K - L(A) - 1\}) \\
&= P2_{s1} \oplus P_{s2} \oplus P_{s3},
\end{aligned}
$$

$\because P_{s2} \neq 0$, according to the associative law,

$$
\begin{aligned}
P1(R) &= P1_{s1} \oplus P_{s2} \oplus P_{s3} = \left(P1_{s1} \oplus P_{s2}\right) \oplus P_{s3} = P_{s2} \oplus P_{s3} \\
P2(R) &= P2_{s1} \oplus P_{s2} \oplus P_{s3} = \left(P2_{s1} \oplus P_{s2}\right) \oplus P_{s3} = P_{s2} \oplus P_{s3} \\
\therefore P1(R) &= P2(R).
\end{aligned}
$$

According to Theorems 1 and 2, $P1 \Leftrightarrow P2$.

If P2 is the election model of P1, we say P2 = Ele(P1). Actually, any node can be elected as a representative, resulting in different compression ratios, and the proof method is similar.



(a) The original trie **(b)** the trie after transformation **(c)** three steps to match

Fig. 4 Representative model

3.3.2 Representative model

Representative: after a successful election, the common ancestor will exercise the right of representative immediately: set the next-hop of its voters (those candidates which own the same next-hop with representative) to 0. As shown in Fig. 4, the next-hop of A and B is same, and A is the nearest ancestor of B. In this case, B's next-hop is set to zero.

Proof ∀IP address R, obviously $L(R) = 32$, $R = [0,1]\{32\}$. Suppose $R = [0,1]\{L(A)\}[0,1]\{L(AB)\}[0,1]\{32 - L(B)\}$,

Step 1 Match $[0,1]\{L(A)\}$

$$
\left.
\begin{aligned}
[0,1]\{L(A)\} = (A) \Rightarrow & \begin{cases} P1([0,1]\{L(A)\}) = P1(A) \\ P2([0,1]\{L(A)\}) = P2(A) \end{cases} \\
[0,1]\{L(A)\} \neq (A) \Rightarrow & \begin{cases} P1([0,1]\{L(A)\}) = P1(\widetilde{A}) \\ P2([0,1]\{L(A)\}) = P2(\widetilde{A}) \end{cases} \\
& P1(A) = P2(A) \\
& P1(\widetilde{A}) = P2(\widetilde{A}A)
\end{aligned}
\right\}.
$$

$$
\Rightarrow P1([0,1]\{L(A)\}) = P2([0,1]\{L(A)\}) = P_{s1}
$$

Step 2 Match $[0,1]\{L(AB)\}$

$$
\left.
\begin{aligned}
[0,1]\{L(AB)\} = (AB) \Rightarrow & \begin{cases} P1([0,1]\{L(AB)\}) = P1(B) \\ P2([0,1]\{L(AB)\}) = P2(A) \end{cases} \\
[0,1]\{L(AB)\} \neq (AB) \Rightarrow & \begin{cases} P1([0,1]\{L(AB)\}) = P1(\widetilde{B}) \\ P2([0,1]\{L(AB)\}) = P2(\widetilde{B}) \end{cases} \\
& P1(B) = P2(A) \\
& P1(\widetilde{B}) = P2(\widetilde{B})
\end{aligned}
\right\}.
$$

$$
\Rightarrow P1([0,1]\{L(AB)\}) = P2([0,1]\{L(AB)\}) = P_{s2}
$$

Step 3 Match $[0,1]\{32 - L(B)\}$

$$\left.\begin{aligned} P1([0,1]\{32-L(B)\}) &= P1(B*) \\ P2([0,1]\{32-L(B)\}) &= P2(B*) \\ P1(B*) &= P2(B*) \end{aligned}\right\} \cdot$$
$$\Rightarrow P1([0,1]\{32-L(B)\}) = P2([0,1]\{32-L(B)\}) = P_{s3}$$

According to Steps 1, 2 and 3, Theorems 1 and 2, we can get P1(R) = P2(R). Therefore, $\forall R,\ P1(R) = P2(R)$ holds according to Theorem 2. $\therefore$ P1 $\Leftrightarrow$ P2. □

If P2 is the representative model of P1, we say P2 = Rep(P1). We insist that all models can be proven by the combination of election model and representative model.

4 The worst case of FIB compression solution

In this section, the bound of the worst case of FIB compression solution is computed, so as to prove the feasibility and effectiveness of FIB compression algorithms.

4.1 Bound of the worst case for full IP address space

4.1.1 Pigeonhole principle

In mathematics, the *Pigeonhole Principle* states that if n + 1 objects are distributed into n boxes, then at least one box contains two or more of the objects (Brualdi 2009). This is a simple but very useful principle. For example, if there are five people from four countries, there are at least two people from the same country.

4.1.2 The worst case for full IP address space[5]

For IPV4, the address space is 2^{32}. Suppose there are 31 ports and 2^{32} prefixes with the length of 32 (full IP address space) in a routing table. At level 32, every 32 nodes elect their common ancestor. At least two ports are the same according to the *Pigeonhole Principle*. Therefore, at least two nodes of 32 nodes can be compressed into one, and thus at least $2^{32}/32 = 2^{27}$ nodes are reduced. At level 27 of the trie, there are 2^{27} nodes. Similarly, 32 nodes select their common ancestor. According to the *Pigeonhole Principle*, at least two nodes can be compressed into one, and $2^{27}/32 = 2^{22}$ nodes are reduced. Therefore, the number of left nodes is at most

[5] In this paper, Full IP Address Space refers to the binary trie whose internal nodes are all pushed to level 32.

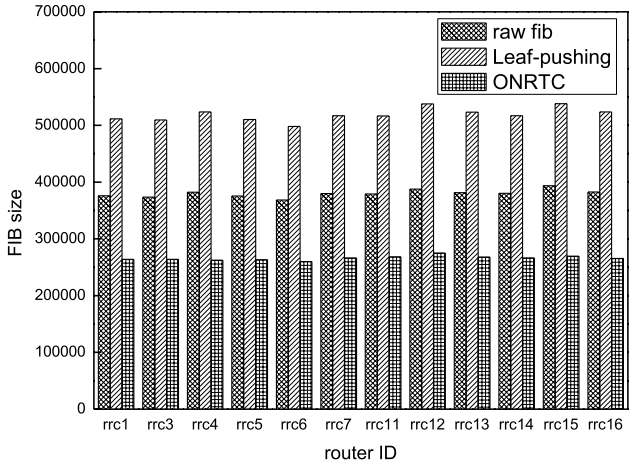


Fig. 5 FIB size comparison using ONRTC and leaf-pushing

$$R = 2^{32} - \frac{2^{32}}{2^{5\times1}} - \frac{2^{32}}{2^{5\times2}} - \frac{2^{32}}{2^{5\times3}} - \cdots - \frac{2^{32}}{2^{5\times6}}. \quad (1)$$

This worst case exists—if the preorder traverse results are N_i (i = 1, 2, 3…), and the next-hop of N_i is represented by $P(N_i)$, which satisfies:

$P(N_i) = i \bmod(32)$.

In this case, the number of compressed routing table by optimal algorithm is R in Eq. (1).

4.2 The worst case for complete binary trie

From above, we can compute the bound of the worst case of FIB compression ratio of Full IP Address Space. However, this is not enough to evaluate the FIB compression ratio in general cases. A more aggressive and general conclusion needs to be drawn.

In order to follow the above method, the trie must be a complete binary trie. Therefore, the actual binary trie used to store routing table should be equally transformed into a complete trie firstly. Fortunately, there are already two algorithms can construct an equivalent complete trie: Leaf-pushing (Srinivasan and Varghese 1999) and ONRTC (Yang et al. 2012) algorithm. These two algorithms are originally to eliminate overlap. Fortunately, the trie after overlap elimination is a complete trie which is equivalent to the original trie for packet forwarding.

In order to evaluate the performance of ONRTC and Leaf-pushing algorithm in an objective and complete way, the RIB packets at 8:00 on August 8 in 2011 from 12 routers at http://www.ripe.net (RIPE Network Coordination Centre) are selected. The results of these two algorithms are shown in Fig. 5. The x-axis of Fig. 5 means the router ID of the 12 routers, and the y-axis means the FIB size of raw FIB,

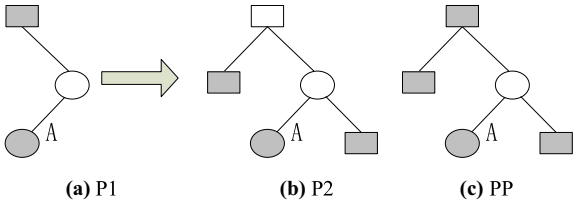


Fig. 6 The model of Leaf-pushing

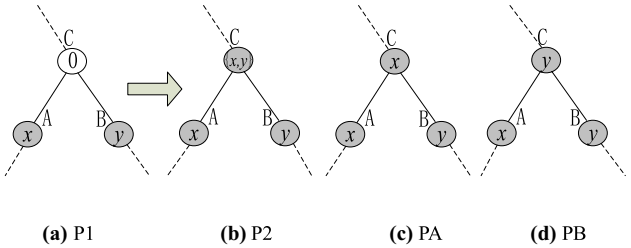


Fig. 7 The model of 'UNION' operation

processed by Leaf-pushing and ONRTC. It can be observed that ONRTC can achieve a much smaller routing table size (Xiao et al. 2018a, b) than Leaf-pushing. ONRTC constructs non-overlap trie, and has been proven to be optimal. Experimental results show that ONRTC can achieve 71% compression ratio in average.

Theorem 3 *As long as the port number n is smaller than M (the prefix number of routing table after compressed by ONRTC), FIB compression solution is feasible, regardless of the distribution of nodes or ports.*

Proof According to the precondition, n < M, as well as the *Pigeonhole Principle*, at least two nodes in the trie after overlap elimination own the same next-hop. Therefore, at least two nodes can be compressed into one. In other words, FIB compression solution works. In conclusion, FIB compression solution is feasible, as long as n < M.

Theorem 4 *To be general, suppose the original routing table size is N. After overlap elimination, the number of leaf nodes is M, suppose* $M = 2^m$ *and the port number is n. Suppose* $n = 2^k - b,\ 0 \le b < 2^k - 2^{k-1},$ *and* $m = ck + s,\ 0 \le s < k$. *Then*

$$R = 2^m - \frac{2^m}{2^{k\times1}} - \frac{2^m}{2^{k\times2}} - \frac{2^m}{2^{k\times3}} - \cdots - \frac{2^m}{2^{k\times c}}. \tag{2}$$

This equation suggests that given the number of routing table size N (or M ≈ 0.71N) and the port number, FIB compression solution can achieve a result of R. The mathematical derivation process is similar with Eq. (1).

To be specific, for example, the size of routing table of RRC 07, which is located in Amsterdam, is 379,685 at 8:00 on August 8 in 2011. After overlap elimination using ONRTC, the size is reduced to 266,338. It means that N = 379,685 and M = 266,338, thus m ≈ 18.02. The next-hop number of RRC 07 is 14, indicating n = 14. Furthermore, according to $n = 2^k - b,\ 0 \le b < 2^k - 2^{k-1}$ and $m = ck + s,\ 0 \le s < k$, We can get $k = 4,\ c = 4$, and finally,

$$\begin{aligned} R &= 2^m - \frac{2^m}{2^{k\times1}} - \frac{2^m}{2^{k\times2}} - \frac{2^m}{2^{k\times3}} - \cdots - \frac{2^m}{2^{k\times c}} \\ &= 266338 * \left(1 - \frac{1}{2^4} - \frac{1}{2^{4\times2}} - \frac{1}{2^{4\times3}} - \frac{1}{2^{4\times4}}\right) \\ &\approx 248582. \end{aligned}$$

This result suggests that FIB algorithm can achieve a compressed routing table size of 248,582 for RRC 07 regardless of the distribution of nodes and next-hop.

5 Application to FIB compression algorithms

We insist that our group theory can be used to prove the correctness of most FIB compression algorithms. To verify this claim, we apply our group theory to four classic FIB compression algorithms: Leaf-pushing algorithm, ORTC algorithm, patent algorithm, and 4-level algorithm in this section as follows.

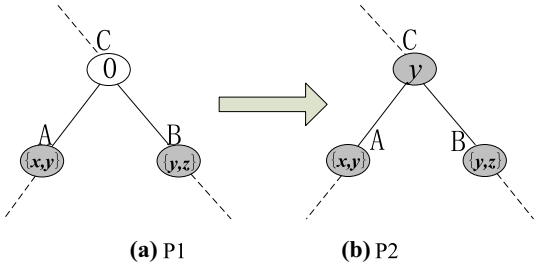
5.1 Application to leaf-pushing algorithm

In Srinivasan and Varghese (1999), leaf-pushing algorithm is proposed to eliminate overlap based on binary trie, but isn't described by models. We map the principle Leaf-pushing algorithm into the following model (see Fig. 6). P1 is the original trie, while P2 is the processed trie using Leaf-pushing, and PP is an auxiliary trie.

Proof

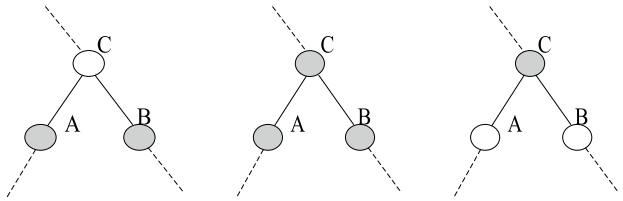
$$\begin{aligned} &\because P1 = Rep(PP),\ \therefore P1 \Leftrightarrow PP; \\ &\because PP = Ele(P2),\ \therefore PP \Leftrightarrow P2; \\ &\qquad\qquad\ \therefore P1 \Leftrightarrow P2. \end{aligned}$$

□



(a) P1 **(b)** P2

Fig. 8 The model of 'AND' operation



(a) P1 **(b)** PP **(c)** P2

Fig. 9 The model of patent algorithm

5.2 Application to ORTC algorithm

ORTC algorithm consists of three Passes. The first Pass 1 is the same as the Leaf-pushing algorithm. The third pass is to delete redundant next hops. The core operations of the ORTC algorithm are the 'UNION' and 'AND' operations during the Pass 2. Therefore, the correctness of 'UNION' and 'AND' is proven in this paper.

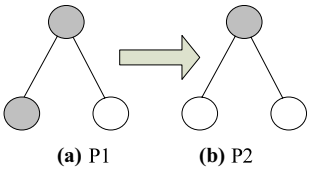
$$A\#B = \begin{cases} A \cup B, & if\ A \cap B \neq \emptyset \\ A \cup B, & if\ A \cap B = \emptyset \end{cases}.$$

Specifically, given two next hop set A and B, if A and B have no next section, then A#B is the union of A and B; otherwise, A#B is the intersection of A and B.

5.2.1 Union (∪)

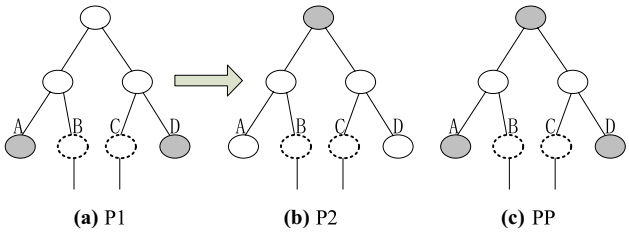
Proof As shown in Fig. 7, P1 is the original trie, while P2 is the processed trie using 'UNION'. PA and PB are two auxiliary tries:

$$\because PA = Ele(P1),\ \therefore PA \Leftrightarrow P1;$$
$$\because PB = Ele(P1),\ \therefore PB \Leftrightarrow P1;$$
$$\therefore P1 \Leftrightarrow PA\ or\ PB \Leftrightarrow P2;$$
$$\therefore P1 \Leftrightarrow P2.$$



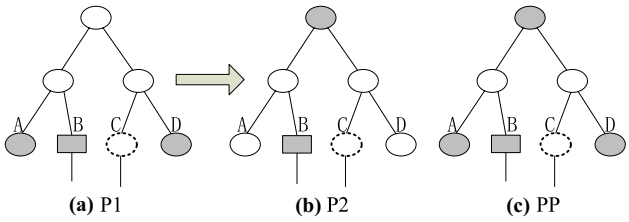
(a) P1 **(b)** P2

Fig. 10 The level 1 model of 4-level algorithm



(a) P1 **(b)** P2 **(c)** PP

Fig. 11 The level 3 model of 4-level algorithm



(a) P1 **(b)** P2 **(c)** PP

Fig. 12 The level 4 model of 4-level algorithm

5.2.2 And ('∩')

Proof As shown in Fig. 8, P1 is the original trie, while P2 is the processed trie using 'AND':

$$\because \text{P2} = \text{Ele(P1)}$$
$$\therefore \text{P1} \Leftrightarrow \text{P2}.$$

5.3 Application to patent algorithm

In Cain (2002), the patent algorithm is proposed. The original implementation of this algorithm uses character matching, which is a waste of time. We map the principle of this algorithm into the following model (see Fig. 9), which achieves the fastest compression speed among all compression algorithms.

Proof As shown in Fig. 9, P1 is the original trie, while P2 is the compression result using patent algorithm, and PP is an auxiliary trie:

$\because \mathrm{PP} = \mathrm{Ele(P1)},\ \therefore \mathrm{PP} \Leftrightarrow \mathrm{P1};$
$\because \mathrm{P2} = \mathrm{Rep(PP)},\ \therefore \mathrm{P2} \Leftrightarrow \mathrm{PP};$
$\therefore \mathrm{P1} \Leftrightarrow \mathrm{P2}.$

5.4 Application to 4-level Algorithm

The 4-level algorithm consists of four transformation models. In order to maintain consistence, the four models are modified to Figs. 10, 11 and 12, but the principle and compression results do not change.

5.4.1 Level 1

Proof As shown in Fig. 10, P1 is the original trie, while P2 is the compression result using Level 1. This compression means that when a prefix node e has the same next hop as its parent or nearest ancestor prefix nodes, node e can be deleted or set to be empty.

$\because P2 = Rep(P1),$
$\therefore P1 \Leftrightarrow P2.$

5.4.2 Level 2

The model of Level 2 is exactly the same as the patent model, and thus the mathematical proof is omitted.

5.4.3 Level 3

Proof As shown in Fig. 11, P1 is the original trie, while P2 is the compressed result using level 3, and PP is an auxiliary trie. Note that those nodes (such as B and C) originally have no next hop, and when they are matched, the packets are usually dropped (Li et al. 2018a, b). To improve the compression ratio, this level compression lets these nodes inherit the next hop of their nearest ancestor nodes. Because in this way, the forwarding behavior of routers is changed, thus such modification is called *weak correctness*. In the model of Fig. 11, the next-hop of node B and C is changed from 0 to 1. It is regarded as non-routable space by 4-level algorithm. If node B and C are not taken into consideration, indicating this only satisfies weak correctness, then

$\because PP = Ele(P1),\ \therefore PP \Leftrightarrow P1;$
$\because P2 = Rep(PP),\ \therefore P2 \Leftrightarrow PP;$
$\therefore P1 \Leftrightarrow P2.$

This proves that level 3 compression satisfies the weak correctness.

5.4.4 Level 4

Proof As shown in Fig. 12, P1 is the original trie, while P2 is the compressed result using level 4, and PP is an auxiliary trie. This level compression is similar to level 3 compression. Obviously, the next-hop of node C is changed from 0 to 1. It is regarded as non-routable space by 4-level algorithm. If node C is not taken into consideration, indicating this only satisfies weak correctness, then

$\because \mathrm{PP} = \mathrm{Ele(P1)},\ \therefore \mathrm{PP} \Leftrightarrow \mathrm{P1};$
$\because \mathrm{P2} = \mathrm{Rep(PP)},\ \therefore \mathrm{P2} \Leftrightarrow \mathrm{PP};$
$\therefore \mathrm{P1} \Leftrightarrow \mathrm{P2}.$

This proves that level 4 compression satisfies the weak correctness. For 4-level algorithm, if the root node is not NULL, i.e., the next-hop of the root node is nonzero, then the Level 3 and Level 4 models achieve no compression. The root node prefix stands for default routing, and thus if and only if there is no default routing in a routing table, Levels 3 and 4 work.

6 Routing table equation test

The mathematical proof method has been elaborated above, but there might be flaws in the process of mathematical derivation and coding. How to guarantee the *ultimate correctness* of these algorithms? The *ultimate correctness* refers to that for any IP address, the compressed routing table tells the same next-hop with the original table. Therefore, we propose routing table equation test (RTET) to judge the equivalence of the two routing tables. RTET firstly builds two tries, then traverses 32-bit IP address space, and compares the next-hop of two tries by using the same IP address. If and only if all are equal, the two routing tables are equivalent. Otherwise, RTET stops and tells the prefix and the different next-hop of the two tries. One comparison of two routing tables by using RTET takes about 16 min. The algorithms (Draves et al. 1999; Cain 2002; Zhao et al. 2010; Li et al. 2011) are all implemented and verified by RTET, using the routing tables downloaded from RIPE Network Coordination Centre.

7 Conclusions

FIB compression has been a hot topic of scientific research for years. There are many FIB compression and overlap elimination algorithms, but there isn't a formal and universal mathematical method to guarantee their correctness. Therefore, we firstly propose a universal mathematical method for trie-transformation algorithms based on a new defined

Group. Secondly, this mathematical proof is applied to several classical trie-transformation algorithms. Furthermore, the following conclusions can be drawn.

- Leaf-pushing algorithm can totally eliminate overlap, but inevitably causing the increase of routing table size.
- Patent algorithm only compresses the two sibling nodes which own the same next-hop, and thus the compression efficiency is less than ORTC and 4-level algorithm.
- 4-level algorithm can achieve a good compression, if and only if the root node is NULL.
- If the NULL root node is considered, ORTC algorithm can achieve a better compression.

Acknowledgements We would like to thank the anonymous reviewers for their thoughtful suggestions. This work is partially supported by Primary Research & Development Plan of China (2018YFB1004403), National Basic Research Program of China (973 Program, 2014CB340405), and NSFC (61672061).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Brualdi, R.A.: Introductory Combinatorics, Chapter 3, 5th edn, pp. 69–70. Machine Press China, Beijing (2009)

Cain, B.: Auto aggregation method for IP prefix/length pairs. http://www.patentgenius.com/patent/6401130.html (2002)

Dai, H., Zhong, Y., Liu, A.X., Wang, W., Li, M.: Noisy bloom filters for multi-set membership testing. In: Proc. ACM SIGMETRICS, pp 139–151 (2016)

Dai, H., Shahzad, M., Liu, A.X., Zhong, Y.: Finding persistent items in data streams. Proc. VLDB Endow. **10**(4), 289–300 (2016)

Dai, H., Meng, L., Liu, A.X.: Finding persistent items in distributed, datasets. In: Proc. IEEE INFOCOM (2018)

Degermaerk, M., Brodnik, A., Carlsson, S., and Pink, S: Small forwarding tables for fast routing lookups. In: Proc. SIGCOMM, NY (1997)

Draves, R., King, C., Venkatachary, S., Zill, B.D.: Constructing optimal IP routing tables. In: Proc. IEEE INFOCOM, pp. 88–97 (1999)

IETF Global Routing Operations (GROW). http://datatracker.ietf.org/wg/grow/charter/ (2015)

IRTF Routing Research Group. https://irtf.org/concluded/rrg (2014)

Korosi, A., Tapolcai, J., Mihálka, B., Mészáros, G., Rétvári, G.: Compressing IP forwarding tables: realizing information-theoretical space bounds and fast lookups simultaneously. In: Proc. IEEE ICNP, IEEE, pp. 332–343 (2014)

Li, D., Cui, H., Hu, Y., et al.: Scalable data center multicast using multi-class bloom filter, network protocols (ICNP). In: 19th IEEE international conference on, pp. 266–275 (2011)

Li, Q., Wang, D., Xu, M., Yang, J.: On the scalability of router forwarding tables: nexthop-selectable FIB aggregation. In: Proc. IEEE INFOCOM (2011)

Li, Q., Xu, M., Chen, M.: NSFIB construction and aggregation with next hop of strict partial order. INFOCOM, 2013, pp. 550–554. In: Proceedings IEEE. IEEE (2013)

Li, F.Z., Xiao, S.W., Pei, T., Li, Jie: Achievable rate maximization for cognitive hybrid satellite-terrestrial networks with af-relays. IEEE J. Sel. Areas Commun. **36**(2), 304–313 (2018)

Li, Z., Chang, B., Wang, S., Liu, A., Zeng, F, Luo, G.: Dynamic compressive wide-band spectrum sensing based on channel energy reconstruction in cognitive internet of things. IEEE Trans. Ind. Inf. (2018)

Lin, D., Zhang, Y., Hu, C., Liu, B., Zhang, X., Pao, D.: Route table partitioning and load balancing for parallel searching with TCAMs. In: Proc. IPDPS (2007)

Liu, Y., Zhao, X., Nam, K., Wang, L., Zhang, B.: Incremental forwarding table aggregation. In: Proc. IEEEE GLOBECOM (2010)

Meng, X., Xu, Z., Zhang, B., Huston, G., Lu, S., Zhang, L.: IPv4 address allocation and the BGP routing table. ACM SIGCOMM Comput Commun Rev **35**, 71–80 (2005)

Nilsson, S., Karlsson, G.: Fast address look-up for internet routers. In: Proceedings of IEEE broadband communications (1998)

Rétvári, G., Tapolcai, J., Kőrösi, A., Majdán, A., Heszberger, Z.: Compressing IP forwarding tables: towards entropy bounds and beyond. ACM SIGCOMM Comput. Commun. Rev. **43**, 111–122 (2013)

RIPE Network Coordination Centre. http://www.ripe.net/data-tools/stats/ris/ris-raw-data

Rottenstreich, O., Radan, M., Cassuto, Y., Keslassy, I., Arad, C., Mizrahi, T., Revah, Y., Hassidim, A.: Compressing forwarding tables. In: Proc. IEEE INFOCOM, IEEE, pp. 1231–1239 (2013)

Ruiz-Sánchez, M.Á., Biersack, E.W., Dabbous, W.: Survey and taxonomy of IP address lookup algorithms. Netw. IEEE **15**, 8–23 (2001)

Srinivasan, V., Varghese, G.: Fast IP lookups using controlled prefix expansion. ACM TOCS **17**, 1–40 (1999)

Vvedensky, D.D.: Group Theory, pp. 14–15. World Scientific Pub., Singapore (2005)

Waldvogel, M., Varghese, G., Turner, J., Plattner, B.: Scalable high speed ip routing lookups. ACM. **27**(4), 25–36 (1997)

Xiao, F., Wang, Z., Ye, N., Wang, R., Li, X.-Y.: One more tag enables fine-grained RFID localization and tracking. IEEE/ACM Trans. Netw. **26**(1), 161–174 (2018a)

Xiao, F., Chen, L., Sha, C., Sun, L., Wang, R., Liu, A.X., Ahmed, F.: Noise tolerant localization for sensor networks. IEEE/ACM Trans. Netw. **26**(4), 1701–1704 (2018b)

Yang, T., Duan, R., Lu, J., Zhang, S., Dai, H., Liu, B.: CLUE: achieving fast update over compressed table for parallel lookup with reduced dynamic redundancy. In: Proc. IEEE ICDCS (2012)

Yang, T., Zhang, T., Zhang, S., Liu, B.: Constructing optimal non-overlap routing tables. Accepted by Proc. IEEE ICC (2012)

Yu, H.: A memory- and time-efficient on-chip TCAM minimizer for IP lookup. DATE '10. In: Proceedings of the conference on design, automation and test in Europe (2010)

Zhao, X., Liu, Y., Wang, L., Zhang, B.: On the aggregatability of router forwarding tables. In: Proc. IEEE INFOCOM (2010)

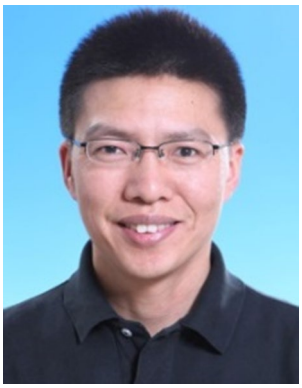
Zheng, K., Hu, C., Lu, H., Liu, B.: A TCAM-based distributed parallel IP lookup scheme and performance analysis. IEEE/ACM Trans. Netw. **14**, 863–875 (2006)

Zhu, H., Xiao, F., Lijuan, S., Wang, R., Yang, P.: R-TTWD: robust device-free through-the-wall detection of moving Human with WiFi. IEEE J. Sel. Areas Commun. **35**(5), 1090–1103 (2017)

Tong Yang received the Ph.D. degree in computer science from Tsinghua University in 2013. He visited the Institute of Computing Technology, Chinese Academy of Sciences, China, from 2013 to 2014. He is currently an Assistant Research Professor with the Computer Science Department, Peking University. His research interests include IP lookups, bloom filters, sketches, and KV stores. He published papers in SIGCOMM, SIGMOD, SIGKDD, VLDB, ATC, INFOCOM, ICDE, ICNP, ICDCS, etc.

Jinyang Li is currently pursuing the bachelor's degree with Peking University, guided by T. Yang. Her research interests include network algorithms, sketches, and Bloom filters.

Chenxingyu Zhao is currently pursuing the bachelor's degree with Peking University, guided by T. Yang. He has published papers in ICDCS. His research interests include network algorithms, sketches, and Bloom filters.



Gaogang Xie received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2002. He is currently a Professor and the Director of the Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include programmable virtual routers, future Internet architecture, and Internet measurement.

Xiaoming Li is currently a Professor in computer science and technology and the Director of the Institute of Network Computing and Information Systems, Peking University, China. He has been leading the effort of developing a Chinese search engine (Tianwang) since 1999. He is also the Founder of the Chinese Web Archive (Web InfoMall). His current research interest is in search engine and Web mining.