

Toward Incremental FIB Aggregation with Quick Selections (FAQS)

Yaoqing Liu
liu@clarkson.edu
Clarkson University

Garegin Grigoryan
gg5996@rit.edu
Rochester Institute of Technology*

Abstract—Several approaches to mitigating the Forwarding Information Base (FIB) overflow problem were developed and software solutions using FIB aggregation are of particular interest. One of the greatest concerns to deploy these algorithms to real networks is their high running time and heavy computational overhead to handle thousands of FIB updates every second. In this work, we manage to use a single tree traversal to implement faster aggregation and update handling algorithm with much lower memory footprint than other existing work. We utilize 6-year realistic IPv4 and IPv6 routing tables from 2011 to 2016 to evaluate the performance of our algorithm with various metrics. To the best of our knowledge, it is the first time that IPv6 FIB aggregation has been performed. Our new solution is 2.53 and 1.75 times as fast as the-state-of-the-art FIB aggregation algorithm for IPv4 and IPv6 FIBs, respectively, while achieving a near-optimal FIB aggregation ratio.

I. INTRODUCTION

A. FIB scalability problem

Several factors contribute to the super-linear growth of global Forwarding Information Base (FIB) size. First of all, the tremendous growth of the number of Internet users results in new network prefixes to be allocated and advertised. Second, network operators often divide large block of IP prefixes allocated to an Autonomous System (AS) into smaller ones and advertise them via Border Gateway Protocol (BGP) to enable fine-grained traffic engineering. According to several research studies ([1], [2]), around 50% of BGP-announced prefixes are more specific prefixes, i.e., the total address space they cover belongs to large address blocks allocated by Internet Assigned Numbers Authority (IANA). 40% of these more specific prefixes are attributed to Traffic Engineering, which is used by network administrators to avoid congested paths [3] or fight against prefix hijacking [4]. Address fragmentation by multi-homing, a practice to connect an end-user network to more than one network in order to provide high throughput and resilient connectivity, is another source of extra prefixes in a routing table ([4], [5]). Overall, the number of entries in FIB has increased almost 40 times since 1994, when the current BGP version 4 emerged. In 2017, the size of FIB has approached 710,000 entries for IPv4 and 40,000 for IPv6, and continues to increase with a super-linear pace [6].

Supporting the current size of FIB and its growth is a challenging task for Default-Free Zone (DFZ) network operators, as they are forced to periodically upgrade their

routing hardware in order to fit the FIB into line cards. It is a heavy financial burden for many small Internet Service Providers (ISPs) to migrate old hardware to new one due to the high costs of line cards and operational expenses ([4], [7]). Some operators avoid upgrading expenses by filtering out specific prefixes with prefix length more than 24, thus affecting the reachability of the Internet [8]. The increasing size of global FIB may also increase chip space for Ternary Content-Addressable Memory (TCAM) design, the Longest Prefix Match (LPM) lookup time [9] and energy consumption by line cards [10].

B. Current approaches

To mitigate the FIB scalability problem, a number of possible solutions were put forward. They can be classified into two broad categories: long-term and short-term solutions. The long-term solutions include revision of the business relations between ASes, e.g., network operators working in the Default-Free Zone (DFZ) can be compensated for keeping all routes in FIB, and re-design of the routing architecture, e.g., splitting address space into a locator (for routing systems) and an identifier (for end systems), may significantly reduce the size of global FIB table, but its wide deployment may take long time [4]. FIB aggregation falls into the category of short-term solutions. Network operators believe it to be one of the most feasible solutions at this moment as it has a clear benefit and many ISPs are seeking such a solution to reduce their operational costs and mitigate their routing scalability problem [4]. FIB aggregation does not require changes on routing hardware and routing architecture, and can be applied locally to each individual router. Several FIB aggregation techniques, such as the Optimal Routing Table Constructor (ORTC) algorithm [11], can greatly reduce the number of FIB entries for an IPv4 FIB by more than 50%. When comparing this result to the rates of FIB growth, we infer that the FIB aggregation may prolong a router's lifetime up to 9 years. However, existing FIB aggregation approaches, such as ORTC-based aggregation algorithms, suffer from a number of challenges that remain to be addressed:

(1) High time costs for processing route updates, including additions, withdrawals and changes. For instance, one of the state-of-the-art FIB aggregation algorithms, *FIFA-S* [12], can achieve optimal aggregation ratio for each update, but needs to perform two subtree traversals in the control plane to update an FIB into aggregated and optimal state.

*This work was done when Garegin Grigoryan was a student at Clarkson University.

(2) Individual routing updates result in a significant number of changes in an FIB, called FIB bursts.

(3) The optimal compression ratio is achieved at the expense of high memory usage: each node generated by the aggregation algorithm in the control plane contains an array of variable size, which stores next-hop candidates to be used for next hop selection for aggregated prefixes.

C. Our contributions

In this work we introduce a new ultra-fast FIB aggregation algorithm: *FIB Aggregation with Quick Selections (FAQS)*. Different from existing aggregation algorithms, *FAQS* uses a single tree traversal to conduct both FIB aggregation and handle FIB updates. It handles routing updates incrementally, without re-aggregation of the whole forwarding table. On a single BGP update, in the worst case, *FAQS* will traverse only the subtree rooted at the updated node and its parents' nodes. Furthermore, unlike *FIFA-S*, *FAQS* keeps only a single next hop at each node and considerably reduces memory consumption for aggregation operations. The outcome of our improvements is the significant acceleration of FIB aggregation and update handling. Although *FAQS* is still a heuristic aggregation algorithm, we experimentally proved its superior performance via multiple realistic datasets in different Routing Information Bases (RIBs) with more than 1 billion route updates from Route Views Project [13] over a 6-year period. The results are briefly described as follows:

(1) *FAQS* achieves high and near-optimal compression ratios: reducing the number of FIB entries by up to 73% for IPv4 and 42% for IPv6.

(2) *FAQS* runs up to 2.53 and 1.75 times as fast as existing *FIFA-S* algorithm for IPv4 and IPv6 FIBs, respectively.

(3) *FAQS* reduces the average number of FIB changes by 30% for IPv4 routing tables and by 10% for IPv6 routing tables.

(4) *FAQS* can save up to 30% of memory consumption compared with *FIFA-S* algorithm that achieves optimal aggregation ratio. *FIFA* series [12] have three algorithms: *FIFA-T*, *FIFA-H* and *FIFA-S*. We did not compare *FAQS* with *FIFA-T* and *FIFA-H* as both of them are threshold-based aggregation algorithms (not strictly incremental). Namely, when a threshold is reached, both algorithms need to re-aggregate over an entire tree/subtree. However, both *FAQS* and *FIFA-S* are strictly incremental FIB aggregation algorithms for every single update.

II. DESIGN

A. FIB aggregation in a nutshell

FIB aggregation refers to a process, that merges two or more FIB entries with different prefixes and same next hop into one. While FIB aggregation may significantly compress the size of an FIB, the aggregation process should not change the forwarding behaviors of any packet. Namely, the next hop for any packet should be same before and after aggregation. In Table Ia, FIB entries *B* and *C* have the same next hop value as the entry *A*, which fully covers IP address blocks of both

TABLE I: FIB aggregation process

(a) Original FIB table

Label	Prefix	Next Hop
A	141.92.0.0/16	1
B	141.92.64.0/18	1
C	141.92.0.0/19	1
D	141.92.192.0/19	2
E	141.92.224.0/19	2

(b) Compressed FIB table

Label	Prefix	Next Hop
A	141.92.0.0/16	1
D	141.92.192.0/19	2
E	141.92.224.0/19	2

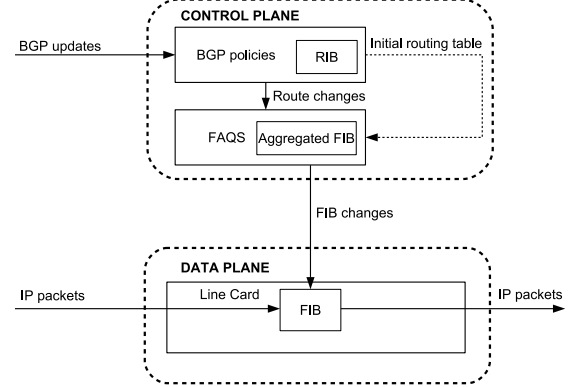


Fig. 1: FAQS Module

B and *C*. Hence, excluding the entries *B* and *C* from the FIB table will not change the forwarding behaviors of any packets matching against *B* or *C*, which preserves the *Forwarding Correctness* rule. Excluding the entries *D* or *E*, in contrast, will not preserve *Forwarding Correctness*, e.g., packets with destination IP addresses from these blocks will be forwarded to the next hop 1 instead of 2. The correctly aggregated FIB is given in Table Ib.

B. FAQS overview

As illustrated in Figure 1, FIB aggregation and *FAQS* algorithm operate in a router's Control Plane, between the RIB and the FIB. When the router boots up, *FAQS* algorithm aggregates the initial set of routes from the RIB and downloads them into the FIB. We call this process *Static FIB Aggregation*. Meanwhile, *FAQS* keeps a copy of the aggregated FIB with various flags to process future route updates. After a routing update, either an addition, a change, or a withdrawal, is advertised via a routing protocol, e.g., BGP, the router first updates its RIB in accordance with BGP decision process. Subsequently, the routing changes are pushed to the aggregation module, where *FAQS* algorithm carries out incremental FIB updates over the aggregated FIB, located in the control plane. A routing update, applied to an aggregated FIB, may not always lead to changes in an FIB. In the meantime, it may result in multiple FIB changes: adding new entries to an FIB, changing next hop values for existing entries or deleting existing entries. If there are FIB changes, *FAQS* installs them in the line cards located

in the data plane.

The remaining part of this section describes both *Static FIB Aggregation* process and *Incremental FIB Update Handling* process in detail.

C. Static FIB aggregation

FAQS uses a data structure based on the PATRICIA trie (PT) [14]. Each node in the PT has the following fields (we assume the current node labeled as n):

(1) *Node type*, denoted by $T(n)$. If a node was derived from an original FIB entry, the value is *REAL*; otherwise, if a PT node is only an ancillary node that helps to form the PT, the value is *FAKE*. In Figure 2(a), $T(F)=T(G)=FAKE$, and $T(A)=T(B)=T(C)=T(D)=T(E)=REAL$.

(2) *Original next hop*. The next hop value that is associated with an original FIB prefix and mapped to a PT node, denoted by $O(n)$. For a *REAL* node, it is taken from the FIB; for a *FAKE* node, it is derived from the original hop of its nearest *REAL* ancestor node during the top-down instantiation described below.

(3) *Selected next hop*. The next hop value of a prefix after aggregation, denoted by $S(n)$. Note that a *selected next hop* may be different from an *original next hop* for the same prefix as long as aggregated FIB has exactly the same forwarding behaviors as the original one.

(4) *FIB status*, denoted by $F(n)$. Indicates whether the prefix and its selected next hop should be placed in the FIB or not after FIB aggregation. $F(n)$ can be equal to *IN_FIB* or *NON_FIB*. All routes with the status $F(n)=IN_FIB$ account for the entire aggregated FIB.

After the initial PT is built from an original FIB, the nodes corresponding to its prefixes have the original next hop from FIB, the *REAL* node type, and an empty selected next hop. The auxiliary nodes in the PT have an empty original next hop, the *FAKE* type, and an empty selected next hop. Starting from here as shown in Figure 2(a), *Static FIB Aggregation* uses one-time post-order traversal to complete the whole aggregation, which consists of a recursive top-down and bottom-up stage.

• **Post-order top-down instantiation for an original next hop:** For simplicity, the root node in the PT has the prefix 0/0 and the *REAL* type. Its original next hop is either derived from the original FIB (if the FIB has a default next hop), or equal to 0 (to indicate a packet drop). From the root node of the PT, we instantiate the original next hop of each *FAKE* node n based on $O(n)=O(n.ancestor)$, where $n.ancestor$ is n 's nearest *REAL* ancestor. Figure 2(b) shows the results after top-down process. The next hops of *FAKE* nodes $O(F)$ and $O(G)$ are derived from the nearest *REAL* ancestor A .

• **Post-order bottom-up assignment for selected next hop and FIB status:** The bottom-up process consists of two operations for each node: assigning a node's selected next hop and determining the FIB status of its children. The selected next hop is assigned as follows:

- I. Leaf nodes: $S(n)=O(n)$.
- II. Internal nodes:

(1) $S(n)=S(n.l)$, when the following conditions are satisfied: $O(n) \neq S(n.r)$, $len(n.l)-len(n)=1$ and $len(n.r)-len(n)=1$, where $n.l$ and $n.r$ are node n 's left and right child, and $len(n)$ represents the length of the prefix on node n . Intuitively, the selected next hop value equals to its left child's selected next hop, when this node has two children nodes and the prefix length differences between this node and both of its children are exactly one, and the right child's selected next hop is different from its own original next hop.

(2) $S(n)=O(n)$ in other cases. There can be three cases: (a) n misses a child node; (b) The length of a child's prefix is longer than that of this node by more than 1; and (c) The selected next hop of a right child equals to the original next hop of n .

The next step for the bottom-up process is determining the FIB status of each node's children nodes. Assume $n.l$ and $n.r$ denote directly connected children of a node n . Then,

(1) $F(n.l)=IN_FIB$, if $n.l$ exists and $S(n.l) \neq S(n)$.

(2) $F(n.r)=IN_FIB$, if $n.r$ exists and $S(n.r) \neq S(n)$.

Otherwise, children's node status will be *NON_FIB*.

Intuitively, we start aggregation from the leaf prefixes and recursively assign selected next hops based on their original next hops. When a child's selected next hop is the same as its parent's, the child's prefix and selected next hop can be excluded from the aggregated FIB. The process stops at the root node, which is always *IN_FIB*. The resultant aggregated FIB will have exactly the same forwarding behaviors as the original one. Figure 2(c) shows the results after the bottom-up process. Algorithms 1, 2, and 3 present the pseudo code for the static FIB aggregation process. Finally, Table II illustrates the aggregated results, where the original five FIB entries are aggregated into two.

Algorithm 1 Static FIB Aggregation

```

1: procedure StaticAggregation(node)
2:    $p \leftarrow node.parent$ 
3:    $l \leftarrow node.left$ 
4:    $r \leftarrow node.right$ 
5:   if  $T(node) \neq REAL$  then
6:      $O(node) \leftarrow O(p)$ 
7:   end if
8:   if  $l \neq NULL$  then
9:     StaticAggregation( $l$ )
10:  end if
11:  if  $r \neq NULL$  then
12:    StaticAggregation( $r$ )
13:  end if
14:   $setSelectedNextHop(node)$ 
15:   $setChildFIBstatus(node)$ 
16: end procedure

```

D. Incremental FIB update handling

FIB updates consist of two categories: (a) Route announcements, including new routes and route changes, and (b) Route withdrawals.

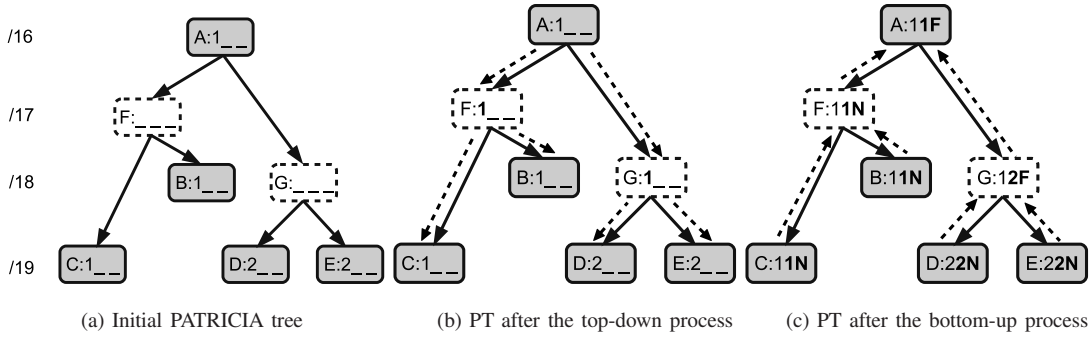


Fig. 2: Static FIB aggregation of FIB from Table Ia.

Fields in a node: (1) original next hop; (2) selected next hop, (3) FIB status: (F:IN_FIB, N:NON_FIB).

The solid nodes denote *REAL* nodes, whose prefixes are from the original FIB.

Algorithm 2 Assignment of Selected Next Hop

```

1: procedure SetSelectedNexthop(node)
2:    $l \leftarrow \text{node}.l$ 
3:    $r \leftarrow \text{node}.r$ 
4:   if  $l \neq \text{NULL} \wedge r \neq \text{NULL} \wedge$ 
        $\text{len}(l) - \text{len}(\text{node}) = 1 \wedge$ 
        $\text{len}(r) - \text{len}(\text{node}) = 1 \wedge$ 
        $O(\text{node}) \neq S(r)$  then
5:      $S(\text{node}) \leftarrow S(l)$ 
6:   else
7:      $S(\text{node}) \leftarrow O(\text{node})$ 
8:   end if
9: end procedure

```

Algorithm 3 Determine FIB status for Children Nodes

```

1: procedure SetChildFIBstatus(node)
2:    $l \leftarrow \text{node}.l$ 
3:    $r \leftarrow \text{node}.r$ 
4:   if  $l \neq \text{NULL}$  then
5:     if  $S(\text{node}) \neq S(l)$  then
6:        $F(l) \leftarrow \text{IN\_FIB}$ 
7:     else
8:        $F(l) \leftarrow \text{NON\_FIB}$ 
9:     end if
10:  end if
11:  if  $r \neq \text{NULL}$  then
12:    if  $S(\text{node}) \neq S(r)$  then
13:       $F(r) \leftarrow \text{IN\_FIB}$ 
14:    else
15:       $F(r) \leftarrow \text{NON\_FIB}$ 
16:    end if
17:  end if
18: end procedure

```

• **Route announcements:** If the announced route is a new route, *FAQS* algorithm generates a *REAL* node with the corresponding original next hop in the PT; if it is a route update, it simply changes the original next hop value accordingly. In order to maintain a good aggregation ratio and forwarding correctness, the aggregated FIB needs to be re-aggregated. In *FAQS*, two portions of the PT may be affected: the subtree rooted at the updated node and the ancestors upon it. Specifically, the original next hop, the selected next hop and the FIB status of each node under the subtree need to be checked and updated if necessary. The process is similar to the procedure of the static FIB aggregation for the entire PT. Also, the selected next hop and the FIB status of each ancestor need to be checked and refreshed if necessary to maintain forwarding correctness. The procedure seems to be tedious, however, we leverage the following three crucial optimization techniques to greatly reduce the overall time costs and memory access times.

(a) When adding a *REAL* node or updating a *FAKE* node, if the original next hop of this node's parent $O(n.\text{parent})$ is same as the new next hop of the updated node $O(n)$, then the top-down process can immediately terminate, since a parent's original next hop in the subtree rooted at n does not change.

(b) Similarly, during the period of updating the subtree, if the node type of a node $T(n)$ is *REAL*, then the top-down process can stop on the current branch, because the original next hop of that node does not change.

(c) During the period of updating the ancestors, if the newly selected next hop of an ancestor n is the same as the old one before the update, then the bottom-up traversal can stop. Since update only happens on one branch and a parent's selected next hop is determined by its children's selected next hop, the preservation of a selected next hop of a node n guarantees the invariance of all nodes above it.

Algorithms 4, 5 and 6 illustrate the whole process of incremental update handling and Figure 3 demonstrates an example to update a route with a new next hop, where the second and third optimization techniques are applied. In the example, Node *D* has an update with a new next hop 3. First the original next hop changes to 3 and other fields are freed; then the update-tree process stops when encountering a *REAL*

TABLE II: FIB entries after Aggregation by *FAQS*

Label	Prefix	Next Hop
A	141.92.0.0/16	1
G	141.92.192.0/18	2

node G . After that, the update-ancestor process stops when the same selected next hop 1 is discovered at node B . As a result, we can observe that only a small portion of the trie has been traversed to incrementally handle the update.

Algorithm 4 Incremental FIB Update Handling

```

1: procedure UpdateNode(prefix, nexthop)
2:   node  $\leftarrow$  find(prefix)
3:   if node = NULL then
4:     node  $\leftarrow$  initialize(prefix, nexthop)
5:     T(node)  $\leftarrow$  REAL
6:     p  $\leftarrow$  node.parent
7:     if O(p)  $\neq$  O(node) then
8:       UpdateSubtree(node)
9:       UpdateAncestors(node)
10:    end if
11:  else
12:    if T(node)  $\neq$  REAL then
13:      T(node)  $\leftarrow$  REAL
14:    end if
15:    if O(node)  $\neq$  nexthop then
16:      O(node)  $\leftarrow$  nexthop
17:      UpdateSubtree(node)
18:      UpdateAncestors(node)
19:    end if
20:  end if
21: end procedure

```

Algorithm 5 Update Subtree

```

1: procedure UpdateSubtree(node)
2:   l  $\leftarrow$  node.l
3:   r  $\leftarrow$  node.r
4:   if l  $\neq$  NULL  $\wedge$  T(l)  $\neq$  REAL then
5:     O(l)  $\leftarrow$  O(node)
6:     UpdateSubtree(l)
7:   end if
8:   if r  $\neq$  NULL  $\wedge$  T(r)  $\neq$  REAL then
9:     O(r)  $\leftarrow$  O(node)
10:    UpdateSubtree(r)
11:  end if
12:  setSelectedNexthop(node)
13:  setChildFIBstatus(node)
14: end procedure

```

• **Route withdrawal:** The *FAQS* algorithm handles the prefix withdrawals within two steps:

(a) Node removal. First, *FAQS* looks up the corresponding *REAL* node from the PT. If the node is found, then *FAQS* checks if it is removable. A removable node refers to a node, which will not affect the PT structure after its deletion. In such case, *FAQS* deletes the node and reorganizes the pointers of its parent and child. Otherwise, if the node is not removable, *FAQS* changes its type to *FAKE* and frees the values of the original next hop, the selected next hop and the FIB status.

Algorithm 6 Update Ancestors

```

1: procedure UpdateAncestors(node)
2:   p  $\leftarrow$  node.parent
3:   while p  $\neq$  NULL do
4:     oldSlctNexthop  $\leftarrow$  S(p)
5:     setSelectedNexthop(p)
6:     setChildFIBstatus(p)
7:     if oldSlctNexthop = S(p) then
8:       break
9:     end if
10:    p  $\leftarrow$  p.parent
11:  end while
12: end procedure

```

(b) Trie update. Starting from the parent node of the deleted or updated node, the incremental update process will be the same as the case of route announcements. First, *FAQS* does a top-down update of the original next hops of nodes on the subtree; next, it bottom-up updates the values of the selected next hops and the FIB status for each node all the way to the point where a new selected next hop does not change. The three optimization techniques used in route announcements apply here as well.

III. EVALUATION

We used realistic IPv4 and IPv6 routing tables from 2011 to 2016 in Route Views project [13] for the evaluation. We collected one baseline routing table on 01/01/2011 for both IPv4 and IPv6, and applied all following updates to obtain the aggregation results. We use AS neighbors as the next hops for FIB tables, because local FIB interface information is not available in the dataset. Normally, the number of interfaces in a FIB is much less than the number of its neighbors. Thus our results underestimate the real FIB aggregation effects. We verified the forwarding behaviors before and after aggregation and they are equivalent. We ran our experiment on an Intel Xeon Processor E5-2603 v3 1.60GHz machine. We compared our *FAQS* algorithm with the optimal ORTC-based *FIFA-S* [12] aggregation algorithm. Unlike *FIFA-T*, a faster version of *FIFA* algorithms, *FIFA-S* has significantly smaller FIB bursts, which is critical since writing operations on TCAM are slow [15].

We used the following metrics for our experiment:

- 1) *FIB Size*: the total number of entries before and after aggregation. *Aggregation Ratio* is calculated by the ratio between the total number of the FIB entries after aggregation and before aggregation.
- 2) *FIB Aggregation Time*: the time spent handling all route updates by the aggregation algorithm (before pushing FIB changes into the data plane).
- 3) *Total Number of FIB Changes*: the total number of FIB changes that are pushed into the data plane by the aggregation module upon handling all route updates. One route update from the control plane may result in zero or more FIB changes to the data plane FIB due to the incremental FIB aggregation process. If there is no

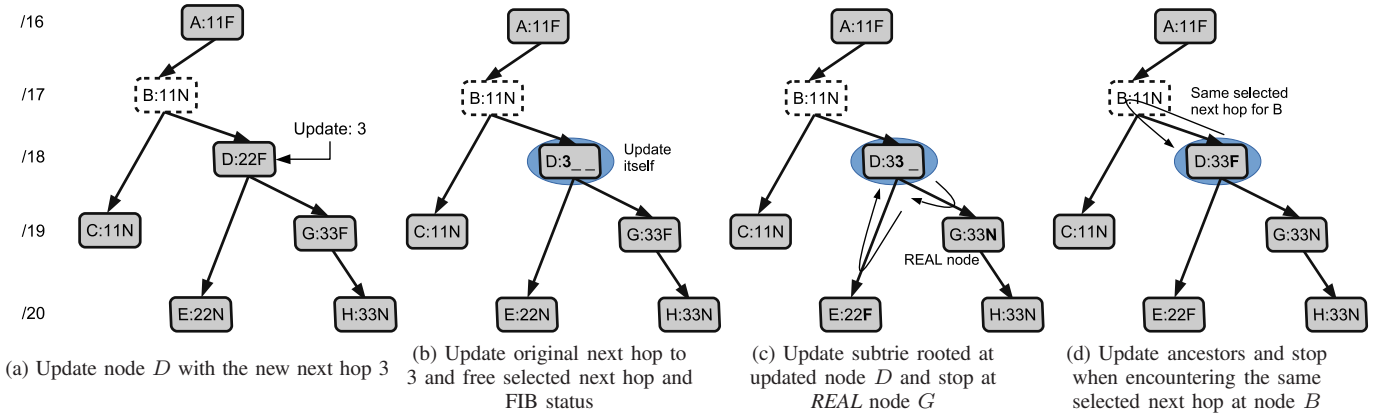


Fig. 3: Incremental FIB update handling by FAQS

aggregation, one route update corresponds to one FIB change.

- 4) *FIB Burst*: The number of FIB changes caused by a single route update, either a route announcement or a withdrawal.

A. IPv4 results

We use five routing tables from different ASes to demonstrate the dependency of aggregation performance on the number of neighbors (i.e. the number of possible next hops). The number of next hops ranges from 21 to 4500. To illustrate the worst case, we use a routing table in AS3356 that has 4500 next hops on 12/31/2016. There are more than 426 million route updates to be handled for the 6-year period.

Figure 4(a) shows the number of FIB entries without aggregation, using *FIFA-S* algorithm and *FAQS* aggregation algorithms. The top green line marked by a triangle represents the FIB size without aggregation. The middle line marked by a rectangle represents the FIB size after *FAQS* and the bottom line represents the FIB size after *FIFA-S*. Both of the aggregation algorithms can compress the original FIB by around 60%. Since *FIFA-S* reaches optimal aggregation ratio for each route update, *FAQS* can achieve near-optimal aggregation ratio.

However, *FAQS* uses much less time to complete the aggregation as shown in Figure 4(b). *FIFA-S* takes around 1000s to finish with an average $2.38\mu s$ per update, while *FAQS* takes about 400s to finish with an average $0.94\mu s$ per update. Thus *FAQS* is 2.53 times faster than *FIFA-S* but bears similar aggregation ratio. The primary reason is that *FIFA-S* needs to traverse a subtree twice to handle an update with additional memory consumption but *FAQS* only needs one-time traversal as described in Section II. The numbers also indicate that *FAQS* can handle more than 1 million updates per second and can be well adopted by Internet backbone routers, given that BGP churn can be up to 500,000 per minute [16].

The smaller number of FIB changes to the FIB, the better performance. Figure 4(c) shows that *FAQS* algorithm generates 31% less number of FIB changes than that of *FIFA-S* algorithm (543,309,259 vs 786,633,132). The average number of FIB changes per update is 1.27 for *FAQS* and 1.84 for

FIFA-S. Both algorithms have similar distribution for the size of FIB bursts as shown in Table III(a). The vast majority of FIB bursts (more than 99.97%) in both algorithms consist of 30 FIB changes and less. The largest FIB burst for *FAQS* is 1443, which is slightly smaller than *FIFA-S* (1496). Nonetheless, the update handling time cost for the largest burst in *FAQS* takes only 30% of running time of *FIFA-S*. Table III(a) presents other evaluation results of FIB aggregation for the five ASes. It is interesting to observe that a good percentage (6.05%-14.91%) of FIB updates result in zero FIB changes (column $n_b=0$).

B. IPv6 results

To the best of our knowledge, this is the first time that IPv6 routing tables have been evaluated for their aggregation results. We aggregated FIB tables from AS 6939 with 3501 next hops. The total number of route updates to be handled is more than 122 million. Figure 5 shows the curves of FIB size, aggregation time and the total number of FIB changes. In Figure 5a, we can observe that the size of IPv6 routing tables has increased dramatically since six years back, when there were only less than 5,000 entries. In the end of 2016, it has been close to 35,000. Due to the small size, the aggregation ratios for both *FAQS* and *FIFA-S* are around 60%, which are not as good as IPv4. Since *FIFA-S* outputs the smallest aggregated FIB, *FAQS*'s aggregation ratio for IPv6 is close to optimal. Remarkably, the running time of *FAQS* is much lower than *FIFA-S* (90s vs 160s in Figure 5b) while they have similar aggregation ratios, which again attributes to the one-time subtree traversal with three important optimization techniques for *FAQS* while *FIFA-S* uses two traversals. Table III(b) demonstrates results for both AS6939 and AS33437. AS33437 has only 7 next hops, thus the aggregation ratio is better (58% vs 56% for *FAQS* and *FIFA-S*, respectively) and the burst size is larger than the one in AS6939, because one update in AS33437 may affect a larger area of next hops.

IV. RELATED WORK

A number of FIB aggregation algorithms have been proposed. We highlight a few of them here. *SMALTA* algorithm [17] uses the binary tree data structure and bases on

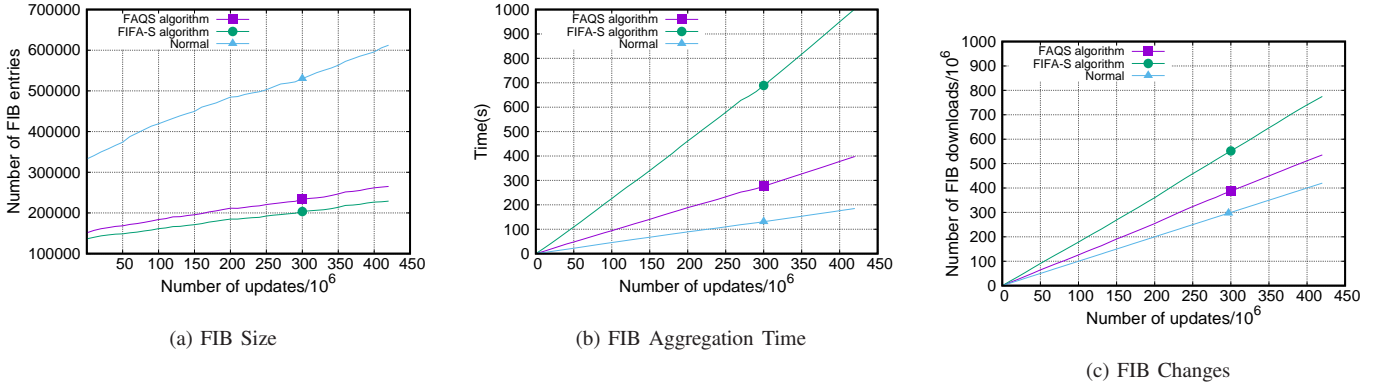


Fig. 4: FIB aggregation of IPv4 routing table (AS 3356)

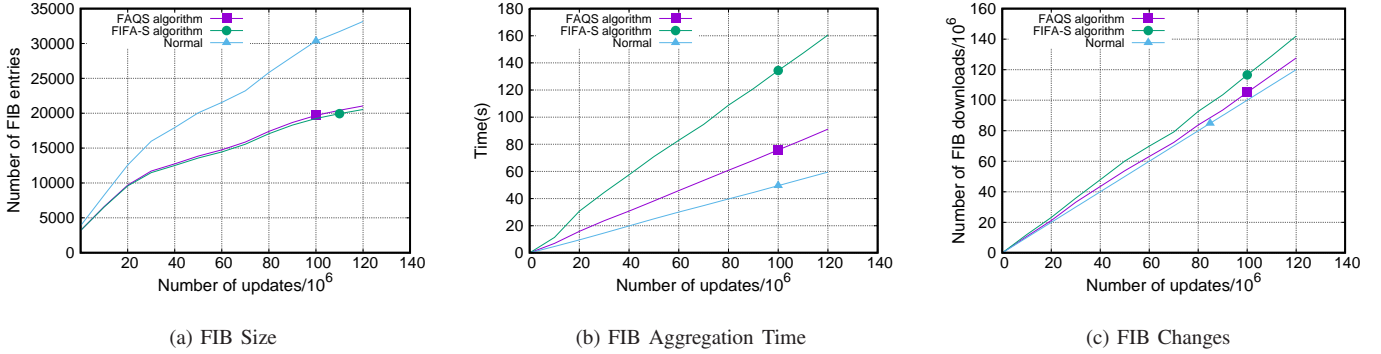


Fig. 5: FIB aggregation of IPv6 routing table (AS 6939)

ORTC algorithm [11], which can achieve one-time optimal aggregation. *SMALTA* takes *ORTC* as the initial FIB aggregation algorithm and processes updates without the optimization of a subtree, rooted at the updated node. Eventually, *SMALTA* requires full re-aggregation of the FIB table upon reaching FIB size threshold. It results in computational spikes and high time costs. In [18], authors study and employ the locality of FIB updates to build Locality-aware FIB Aggregation (LFA) algorithm. In LFA, reaggregation for an updated prefix region is delayed until it is stabilized. However, such approach requires timers attached to nodes which may significantly complicate its operation in the real routers. Bienkowski et al. [19] present a formal study on the trade-off between FIB aggregation and update bursts. In addition, paper presents the algorithm *HIMS* that attaches time-dependent counters to each node as well. However, the paper provides no information on the performance of the algorithm when processing real network routing data. In [20], authors propose MMS, the Memory Management System designed to prolong the lifetime of legacy routers in an ISP. MMS uses parallelization of *ORTC* and can aggregate routing tables locally or on an AS-level. Moreover, MMS may change the forwarding behavior of routers in order to gain additional compression.

Some FIB compression work uses smart data structures to minimize storage size of FIB [21]. In [22], authors present a tunable aggregation algorithm with compressed prefix trees. By changing the deepness of the compression, network operators can manage the trade-off between the aggregation ratio and BGP update overhead. Similarly, Yang et al. in [23]

present two algorithms, *EAP-slow* and *EAP-fast* and compare it with *ORTC*. In [24], authors propose an aggregation algorithm for OpenFlow flow tables using prefix wildcards. FIB aggregation scheme, that applies multiple selectable next hops, is proposed by Li et al. [8]. Abraham et al. [25] create a virtual network system to implement and study FIB aggregation. It is a reusable framework to test the performance of FIB aggregation algorithms in a realistic environment.

Aggregation algorithms such as *Level-1* and *Level-2* [26] compress FIB quickly but bear costly update handling operations. In 2013, Liu et al. developed *FIFA* algorithms [12], which improves *ORTC* algorithm by applying *PATRICIA* trie (PT) with incremental FIB aggregation features.

Our work, *FAQS* algorithm, makes a good balance of aggregation time, ratio and memory consumption. It sacrifices very little aggregation ratio compared with the optimal solution, but speeds up the aggregation more than twice with much less memory consumption. Considering the real-time and efficiency requirements of FIB aggregation, our approach is superior to the existing algorithms.

V. CONCLUSION

FIB Aggregation with Quick Selections (FAQS) is a new FIB aggregation algorithm, that leverages compact data structures and three unique optimization techniques to quickly and incrementally select next hops when handling route updates. As a result, *FAQS* can run up to 2.53 and 1.75 times faster for IPv4 and IPv6, respectively, than the optimal FIB aggregation algorithm while achieving near-optimal aggregation ratio. Mean-

AS	n_u	p_{avg}	Algorithms	r	n_c/n_u	t_{aggr}	t_{peak}	$n_b = 0$	$n_b = 1$	$n_b \leq 30$	b_{max}	m
3356	426551755	3746	FAQS	0.43	1.27	0.94 μ s	3.09ms	12.75%	56.21%	99.97%	1443	175MB
			FIFA	0.37	1.84	2.38 μ s	10.29ms	12.85%	64.23%	99.98%	1496	228MB
7018	782293331	2397	FAQS	0.41	1.21	0.94 μ s	3.09ms	16.43%	61.71%	99.99%	1854	175MB
			FIFA	0.34	1.78	2.06 μ s	8.39ms	16.38%	54.57%	99.97%	1892	229MB
8492	1037150247	1126	FAQS	0.39	1.37	0.93 μ s	3.27ms	6.05%	69.45%	99.97%	4268	178MB
			FIFA	0.32	1.90	2.33 μ s	12.14ms	6.40%	63.77%	99.97%	4657	233MB
1239	295214072	739	FAQS	0.42	1.28	1.09 μ s	3.56ms	13.18%	62.18%	99.98%	1585	175MB
			FIFA	0.36	1.91	2.69 μ s	12.68ms	13.38%	55.03%	99.97%	1952	229MB
3130	402445005	23	FAQS	0.27	1.23	0.95 μ s	3.26ms	14.69%	63.50%	99.98%	6464	174MB
			FIFA	0.20	1.68	2.04 μ s	16.02ms	14.91%	56.74%	99.98%	5524	228MB

(a) IPv4 routing tables

AS	n_u	p_{avg}	Algorithms	r	n_c/n_u	t_{aggr}	t_{peak}	$n_b = 0$	$n_b = 1$	$n_b \leq 30$	b_{max}	m
6939	122903741	2725	FAQS	0.63	1.06	0.76 μ s	1.27ms	7.08%	84.19%	99.99%	181	11MB
			FIFA	0.61	1.18	1.33 μ s	2.97ms	7.09%	81.19%	99.98%	258	14MB
33437	33486605	7	FAQS	0.58	0.98	0.90 μ s	1.42ms	17.47%	73.68%	99.99%	2447	11MB
			FIFA	0.56	1.11	1.43 μ s	2.48ms	17.46%	68.33%	99.99%	2432	14MB

(b) IPv6 routing tables

TABLE III: Evaluation summary (2011-2016 period). n_u - the number of FIB updates; p_{avg} - average peer number; r - aggregation ratio; n_c/n_u - the ratio between the number of FIB changes and FIB updates; t_{aggr} - average aggregation time per update; t_{peak} - peak aggregation time; n_b - percentage of updates with burst values 0, 1 and below 30; b_{max} - maximum burst value; m - memory consumption.

while, it consumes much less memory and generates much smaller number of FIB changes when carrying out frequent updates. The performance enhancement of the new algorithm addresses many concerns from ISPs regarding performance issues, and enhances the probability to push FIB aggregation techniques further to the level of production adoption by the industry.

REFERENCES

- [1] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang, "IPv4 address allocation and the BGP routing table evolution," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, pp. 71–80, 2005.
- [2] This lists the "Top 30" ASes who if they aggregated their announced prefixes could make a significant contribution to the reduction of the current size of the Internet routing table. [Online]. Available: <https://www.cidr-report.org/as2.0/#Aggs>
- [3] F. Valera, I. Van Beijnum, A. Garcia-Martinez, and M. Bagnulo, "Multipath BGP: motivations and solutions," *Next-Generation Internet*, p. 238, 2011.
- [4] X. Zhao, D. J. Pacella, and J. Schiller, "Routing scalability: an operator's view," *IEEE Journal on Selected Areas in communications*, vol. 28, no. 8, pp. 1262–1270, 2010.
- [5] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," *Computer Networks*, vol. 45, no. 1, pp. 45–54, 2004.
- [6] Active BGP entries (FIB). [Online]. Available: <http://bgp.potaroo.net/as6447/>
- [7] V. Khare, D. Jen, X. Zhao, Y. Liu, D. Massey, L. Wang, B. Zhang, and L. Zhang, "Evolution towards global routing scalability," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1363–1375, 2010.
- [8] Q. Li, M. Xu, D. Wang, J. Li, Y. Jiang, and J. Yang, "Nexthop-selectable FIB aggregation: An instant approach for internet routing scalability," *Computer Communications*, vol. 67, pp. 11–22, 2015.
- [9] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," *RFC 4984*, 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4984.txt>
- [10] M. Moradi, F. Qian, Q. Xu, Z. M. Mao, D. Bethea, and M. K. Reiter, "Caesar: High-speed and memory-efficient forwarding engine for future internet architecture," in *Architectures for Networking and Communications Systems (ANCS)*, 2015 ACM/IEEE Symposium on. IEEE, 2015, pp. 171–182.
- [11] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, "Constructing optimal IP routing tables," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 88–97.
- [12] Y. Liu, B. Zhang, and L. Wang, "Fifa: Fast incremental FIB aggregation," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 1–9.
- [13] Advanced Network Technology Center and University of Oregon, "The RouteViews project." [Online]. Available: <http://www.routeviews.org/>
- [14] D. R. Morrison, "PATRICIA - practical algorithm to retrieve information coded in alphanumeric," *Journal of the ACM (JACM)*, vol. 15, no. 4, pp. 514–534, 1968.
- [15] R. Bifulco and A. Matsiuk, "Towards scalable sdn switches: Enabling faster flow table entries installation," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 343–344, 2015.
- [16] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "Bgp churn evolution: a perspective from the core," *IEEE/ACM Transactions on Networking (ToN)*, vol. 20, no. 2, pp. 571–584, 2012.
- [17] Z. A. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis, "SMALTA: practical and near-optimal FIB aggregation," in *Proc. CoNEXT*, 2011.
- [18] N. Sarrar, R. Wuttke, S. Schmid, M. Bienkowski, and S. Uhlig, "Leveraging locality for fib aggregation," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 1930–1935.
- [19] M. Bienkowski, N. Sarrar, S. Schmid, and S. Uhlig, "Competitive FIB aggregation without update churn," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 607–616.
- [20] E. Karpilovsky, M. Caesar, J. Rexford, A. Shaikh, and J. Van Der Merwe, "Practical network-wide compression of ip routing tables," *IEEE Transactions on Network and Service Management*, vol. 9, no. 4, pp. 446–458, 2012.
- [21] G. Rétfári, Z. Csernátó, A. Körösi, J. Tapolcai, A. Császár, G. Enyedi, and G. Pongrácz, "Compressing ip forwarding tables for fun and profit," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012, pp. 1–6.
- [22] G. Rétfári, J. Tapolcai, A. Körösi, A. Majdán, and Z. Heszberger, "Compressing ip forwarding tables: towards entropy bounds and beyond," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 111–122.
- [23] T. Yang, B. Yuan, S. Zhang, T. Zhang, R. Duan, Y. Wang, and B. Liu, "Approaching optimal compression with fast update for large scale routing tables," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*, ser. IWQoS '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 32:1–32:9. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2330748.2330780>
- [24] S. Luo, H. Yu, and L. Li, "Practical flow table aggregation in sdn," *Computer Networks*, vol. 92, pp. 72–88, 2015.
- [25] J. P. Abraham, Y. Liu, L. Wang, and B. Zhang, "A flexible quagga-based virtual network with FIB aggregation," *IEEE Network*, vol. 28, no. 5, pp. 47–53, 2014.

- [26] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the Aggregatability of Router Forwarding Tables," in *Proc. IEEE INFOCOM*, 2010.