NSFIB Construction & Aggregation with Next Hop of Strict Partial Order

Qing Li, Mingwei Xu, Meng Chen
Department of Computer Science & Technology of Tsinghua University
Tsinghua National Laboratory for Information Science and Technology

***Abstract*—The Internet global routing tables have been expanding at a dramatic and increasing rate. In this paper, we propose the next hop of strict partial order to construct and aggregate the Nexthop-Selectable FIB (NSFIB). We control the path stretch caused by NSFIB aggregation by setting an upper limit number of next hops. According to our simulation, our aggregation algorithms shrink the FIB to 5-15%, compared with 20-60% of single-nexthop FIB aggregation algorithms; our method works very well in controlling the path stretch.**

I. Introduction

The report from the Internet Architecture Board (IAB) reveals that Internet routing is facing severe scalability problem with the ever-increasing users [1]. The global routing tables have been expanding at a dramatic and increasing rate for recent years. By July 2012, the routing table size has reached 433,653 (from 15,100 of 1994) [2].

The most urgent requirement for ISPs is shrinking the Forwarding Information Base (FIB), which is the most expensive part of the router memory. Therefore, as an immediate solution, FIB aggregation is proposed. The most significant advantage of FIB aggregation is its supporting of the router-level incremental deployment, which means it can mitigate the pressure of ISPs immediately. However, current single-nexthop FIB aggregation solutions [3–6] cannot provide a satisfactory aggregation performance. Besides, they have a common problem: the performance degrades as the degree of the network topology increases [7].

We proposed Nexthop-Selectable FIB (NSFIB) aggregation in [7]. In NSFIB aggregation, multiple selectable next hops are constructed for each prefix. Two numerically matching prefixes can be aggregated into one only if they share one common next hop. As the possibility of two prefixes sharing one common selectable next hop is greater than that of two prefixes having the same optimal next hop, Although NSFIB aggregation provides another possible path to follow towards a scalable Internet, there are still some problems to address, *e.g.*, an effective NSFIB construction algorithm, the compatibility with other aggregation approaches [3–6] and a method of controlling the path stretch.

In this paper, first, we propose the next hop of strict partial order (SPO next hop) to construct the NSFIB. According to our construction method, the routers on the forwarding path have a relation of strict partial order. Therefore, no forwarding loop exists after the aggregation. Then, we provide an effective algorithm (*SPO-Nexthop*) to compute all the SPO next hops for each destination. By a novel approach of changing the link costs between the source router and its neighbors to ϵ (arbitrarily small quantity), we control the complexity of *SPO-Nexthop* as the same as Shortest Path First (SPF) algorithm, which means that the NSFIB can be re-constructed effectively when the topology changes. Besides, we prove that NSFIB-based aggregation can avoid performance degrading in denser networks, which exists in single-nexthop FIB aggregation.

Then, we provide the algorithms of applying different single-nexthop FIB aggregation approaches [3–6] on NSFIB. We select two typical ones to demonstrate the algorithms: 1) according to the four levels of single-nexthop aggregation algorithms in [4], we propose four levels of NSFIB-based aggregation algorithms; 2) we propose the algorithm of NSFIB-based Optimal Routing Table Construction (NSFIB-ORTC), which computes the minimized aggregated FIB for a given NSFIB, just as ORTC [3] does on the single-nexthop FIB. Although ORTC algorithm computes the minimized aggregated FIB for a NSFIB, the computation involves too much memory overhead. The space complexity of ORTC is $O(W|P|)$ where W = 32 for IPv4 (W = 128 for IPv6) and P is the prefix set of the FIB. Therefore, in the algorithm of NSFIB-ORTC, we employ the path-compressed trie instead of the binary trie, which reduces the space complexity to $|P|$.

After that, we provide the method of setting an upper limit of selectable next hops to control the path stretch caused by NSFIB-based algorithms. The next hops leading to higher path stretch will be filtered according to the upper limit. This provides an efficient way for a router to make a trade-off between the aggregated FIB size and the path stretch.

Finally, we demonstrate the performance of NSFIB aggregation algorithms by comprehensive simulations on China Education and Research Network (CERNET) [8], Rocketfuel [9] topologies and BRITE-generated topologies [10]. The results show: 1) NSFIB-based aggregation algorithms perform better than the corresponding single-nexthop FIB aggregation algorithms. NSFIB Level1, NSFIB Level2 and NSFIB-ORTC can respectively shrink the FIB to about 11%, 9% and 5%, compared with 60%, 30% and 25% of single-nexthop FIB aggregation algorithms. 2) The path stretch ratios of NSFIB aggregation are less than 5% in all the six Rocketfuel topologies. While in denser BRITE topologies, the path stretch ratios are more than 30%. Our method of controlling path stretch works well in these denser topologies.

The research is supported by National Natural Science Foundation of China (61073166 and 61133015), National Basic Research Program of China (973 Program) under Grant 2009CB320502, National High-Tech Research and Development Program of China (863 Program) under Grant 2011AA01A101 and National Sci. & Tech. Pillar Program of China under Grant 2011BAH19B01.

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

II. Background and Related Works

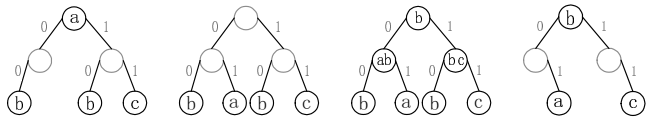
We use a $0/1$ string appended with $*$ to denote a prefix. For example, $10*$ is the prefix that covers all IP addresses starting with the bits 10. A FIB is a set of FIB entries like $< p, h >$, where p is the prefix and h is the corresponding next hop. We organize a FIB as a radix tree (or a trie), following the convention. A trie is an ordered tree structure for quick searching. In most routers, the trie is widely used for organizing the routing tables because it well supports the longest prefix matching principle.

A. Single-Nexthop FIB Shrinking Schemes

In this paper, we refer to the FIB where each prefix has one single next hop as the single-nexthop FIB. There are several aggregation algorithms based on the single-nexthop FIB.

1) General Single-Nexthop FIB Aggregation: General Single-Nexthop FIB aggregation (General FIB Aggregation for short) is the simplest form of FIB aggregation, which is also summarized as *Level-1 aggregation* in [4]. The principle of General FIB Aggregation has been widely used for routing table aggregation since Classless Inter-Domain Routing (CIDR) [11] was introduced. In general FIB aggregation, the prefixes sharing the same next-hop with their immediate ancestor prefixes are aggregated. For example, there are two entries $< 10*, a >$ and $< 101*, a >$ in the FIB. The latter can be aggregated into the former.

2) Optimal Routing Table Constructor: To further shrink the single-nexthop FIB, Draves, *et al.*, proposed *Optimal Routing Table Constructor* (ORTC) [3]. ORTC produces the minimized FIB that preserves the equivalent forwarding behavior by the following three steps. **Step 1**: *"Normalize"* the trie of the FIB so that each inner node has two children. **Step 2**: Calculate the most prevalent next hops at every level of the trie in post order. **Step 3**: Aggregate the prefixes sharing common next hops with their immediate ancestor prefixes from top to down. Fig. 1 shows an example of these three steps. Note that the FIB in Fig. 1(a) cannot be shrunk by General FIB aggregation. However, ORTC can still shrink this FIB from four entries to three entries.

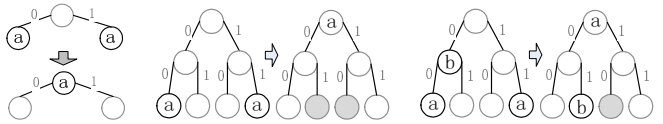


(a) A FIB (b) Step 1 (c) Step 2 (d) Step 3

Fig. 1. The steps of Optimal Routing Table Constructor (ORTC).

3) Level 1-4 Aggregation: Level 1-4 aggregations [4] were proposed to provide different levels of FIB aggregation ISPs with different requirements can select proper levels to deploy. Level 1 aggregation is General FIB Aggregation.

Different from ORTC, Level 2-4 aggregations [4] concentrate on specific sub-tries (part of the prefix space), which can be aggregated by packing some new prefixes; and they cannot guarantee the global optimality. As the specific sub-tries all have limited size (three layers at most), the influence of aggregation is localized at the corresponding part of the whole FIB trie. Thus, any update upon this special sub-trie is also localized, which guarantees a constant update complexity. Fig. 2 shows examples of Level 2-4 aggregations.



(a) Level 2 (b) Level 3 (c) Level 4

Fig. 2. Different levels of FIB aggregation in [4]. Each binary tree represents a sub-trie (of part) of the FIB. In (b) and (c), some extra routable spaces (filled nodes) are introduced.

B. Nexthop-Selectable FIB Aggregation

In [7], we proposed Nexthop-Selectable FIB (NSFIB) aggregation to further compress FIB by constructing multiple selectable next hops for each prefix. A NSFIB is a set of entries, like a FIB, except that i each prefix has multiple selectable next hops (not just the optimal one). In NSFIB aggregation, a covered prefix sharing one common next hop with the covering prefix can be aggregated. For example, $< 101*, \{a, b\} >$ can be aggregated by $< 10*, \{b\} >$.

However, as a first-step work, NSFIB aggregation [7] has several problems to be solved before practical deployment:

- An efficient NSFIB construction algorithm. Inevitable network failures will trigger NSFIB re-construction, therefore, an efficient NSFIB construction algorithm is required.
- The algorithms of applying different single-nexthop FIB aggregation methods [3–6] on the NSFIB.
- An approach to control the path stretch caused by NSFIB aggregation, which might use non-optimal next hops.

III. Construction of Nexthop-Selectable FIB

A. Next Hop of Strict Partial Order

Consider an undirected, weighted topology $G(V, E)$. $\forall R_x \in V$, let $ID(R_x)$ (an 32-bit unsigned integer) be the ID of node R_x. $\forall R_s, R_d \in V$, let $dist(R_s, R_d)$ be the optimal distance from R_s to R_d. For each destination $R_d \in V$, let $(\prec_{R_d})$ be a relation on V such that: $R_a \prec_{R_d} R_b$ iff $dist(R_a, R_d) < dist(R_b, R_d)$, or $dist(R_a, R_d) = dist(R_b, R_d)$ and $ID(R_a) < ID(R_b)$. Note that $(\prec_{R_d})$ is a strict partial order on V.

Definition 1. *Next hop of strict partial order (SPO next hop): R_n is a SPO next hop from R_s to R_d iff R_n is a neighbor of R_s and $R_n \prec_{R_d} R_s$.*

Theorem 1. *Forwarding by the SPO next hop guarantees no routing loop.*

Proof: Suppose not: after some forwarding steps by SPO next hop, the packet loops back to the source router R_s. Note that $R_s \prec_{R_d} R_1 \prec_{R_d} R_2 ... \prec_{R_d} R_s$, where R_1, R_2, *et al.*, are the intermediate forwarding nodes before the packet loop back to R_s. According to the transitive property of $\prec_{R_d}$, $R_s \prec_{R_d} R_s$, which is controversial with the definition $\prec_{R_d}$. Hence, the supposition is false and the theorem is true. ■

NSFIB Construction Method: In Border Gateway Protocol (BGP), each prefix has one optimal route that has an intra-domain destination, *i.e.*, the egress BGP router. A router constructs the Nexthop-Selectable FIB by the following steps:

1) Compute the SPO next hops for all the egress BGP routers based on OSPF;
2) For each prefix p, allocate the SPO next hops to p according to its egress BGP router.

Such that, NSFIB is constructed. After NSFIB aggregation, each packet will be forwarded by one certain SPO next hop, which guarantees a loop-free route towards the egress router.

B. Algorithm for SPO Next Hop Computation

In this section, we provide the efficient algorithm for a router R_s to compute the SPO next hops to intra-domain destination R_d. For each neighbor (R_i) of R_s, $dist(R_i, R_d)$ is required to determine whether R_i is a SPO next hop to R_d. Therefore, R_s have to compute the Shorted Path Tree (SPT) rooted at each neighbor. Extra K (the number of neighbors) times of Shortest Path First (SPF) are required, which is a nontrivial burden for larger and denser topologies.

The algorithm of incremental Shortest Path First (iSPF) [12] is used for re-computing the new SPT based on the old SPT when part of the topology changes, *e.g.*, the cost of some link increases or decreases. Instead of computing a new SPT from scratch, iSPF reduces the computation overhead by adjusting the old SPT to form a new one. We design a novel iSPF-based algorithm, SPO-Nexthop(), to compute the SPO next hops for all the intra-domain destinations.

Algorithm 1, SPO-Nexthop(), computes the SPO next hops to each intra-domain router for R_s with the input of R_s-rooted SPT. Assume R_s has K neighbors $\{R_i | i \in [1, K]\}$. Let $ID(R_x)$ be the ID of the router R_x. Let $cost(R_x, R_y)$ be the cost of the unidirectional link from R_x to R_y. Let ϵ be an arbitrarily small positive value (smaller than any link cost). For each neighbor R_i, SPO-Nexthop() finds the set of routers, to which R_i is a SPO next hop. Totally, K rounds of iSPF are required to construct the unidirectional next hops for all the intra-domain routers.

Theorem 2. *SPO-Nexthop() computes all the unidirectional next hops for each intra-domain destination.*

Proof: We first prove that for any node (R_d) belonging to R_i-rooted subtree in the new SPT, R_i is a SPO next hop from R_s to R_d. Note that two cases exist: $ID(R_s) < ID(R_i)$ or $ID(R_s) > ID(R_i)$.

- Case One: $ID(R_s) < ID(R_i)$, and $Cost(R_s, R_i) = +\epsilon$. $dist(R_i, R_d) < dist(R_s, R_d)$ (*i.e.*, $R_i \prec_{R_d} R_s$) because $dist(R_i, R_d) + \epsilon \leq dist(R_s, R_d)$. Therefore, R_i is a SPO next hop from R_s to R_d.
- Case Two: $ID(R_s) > ID(R_i)$, and $Cost(R_s, R_i) = -\epsilon$. $dist(R_i, R_d) \leq dist(R_s, R_d)$ (*i.e.*, $R_i \prec_{R_d} R_s$) because $dist(R_i, R_d) - \epsilon \leq dist(R_s, R_d) \Rightarrow dist(R_i, R_d) - dist(R_s, R_d) \leq \epsilon \Rightarrow dist(R_i, R_d) - dist(R_s, R_d) \leq 0$. Therefore, R_i is a SPO next hop from R_s to R_d.

We then prove that for any node (R'_d) not belonging to R_i-rooted subtree in the new SPT, R_i is not a SPO next hop from R_s to R_d. There are also two cases:

- Case One: $ID(R_s) < ID(R_i)$, and $Cost(R_s, R_i) = +\epsilon$. $dist(R_i, R'_d) \geq dist(R_s, R'_d)$ because $dist(R_i, R'_d) + \epsilon \geq dist(R_s, R'_d) \Rightarrow dist(R_s, R'_d) - dist(R_i, R'_d) \leq \epsilon \Rightarrow dist(R_s, R'_d) - dist(R_i, R'_d) \leq 0$. Therefore, R_i is not a SPO next hop from R_s to R'_d.
- Case Two: $ID(R_s) > ID(R_i)$, and $Cost(R_s, R_i) = -\epsilon$. $dist(R_i, R'_d) > dist(R_s, R'_d)$ because $dist(R_i, R'_d) - \epsilon \geq dist(R_s, R'_d)$. Therefore, R_i is not a SPO next hop from R_s to R'_d.

Consequently, after K rounds, SPO-Nexthop() finds the SPO next hops to all intra-domain routers. ■

Algorithm 1 SPO-Nexthop(R_s-rooted SPT)

1: **for all** R_i in the neighbor set of R_s **do**
2: If $ID(R_s) < ID(R_i)$, change $cost(R_s, R_i)$ to $+\epsilon$; else, change $cost(R_s, R_i)$ to $-\epsilon$ ($cost(R_i, R_s)$ is not changed);
3: Compute the new SPT by iSPF with the inputs of the original R_s-rooted SPT and the above change;
4: For any node (R_d) belonging to R_i-rooted subtree in the new SPT, add R_i as a SPO next hop of R_d.
5: Restore $cost(R_s, R_i)$ to the original value.
6: **end for**

IV. NSFIB-Based Aggregation Algorithms

Let a FIB be a set $\mathcal{F} = \{\langle p, a_p \rangle | p \in P\}$ where P is the set of prefixes in the FIB and a_p is the next hop for prefix p. Let a Nexthop-Selectable FIB (NSFIB) be a set $\mathcal{F}_{ns} = \{\langle p, A_p \rangle | p \in P_{ns}\}$ where P_{ns} is the prefix set and A_p is the set of selectable next hops for p.

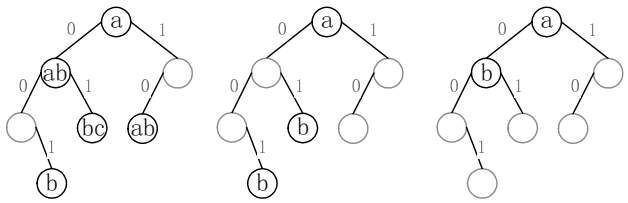
A. Level-1 NSFIB Aggregation

We first propose the algorithm of Level-1 NSFIB Aggregation corresponding to the general FIB aggregation (or Level-1 FIB Aggregation in [4]). *Level-1 NSFIB Aggregation Principle*: the prefixes sharing one common next-hop with their immediate ancestor prefixes are aggregated.

There are two different methods to implement Level-1 NSFIB Aggregation, which are described as follows:

1) The breadth-first greedy method: 1) let p be current unprocessed prefix in $\mathcal{F}_{ns}$, choose x ($\in A_p$) that can aggregate the maximum descendant prefixes; 2) mark $< p, x >$ as an aggregated entry and remove all the descendant prefixes that can be aggregated by the entry of $< p, x >$; 3) do 1) and 2) iteratively in breadth-first order till all the prefixes are processed. Fig. 3(b) shows an aggregated FIB by this algorithm with the NSFIB of Fig. 3(a) as the input.
2) The bottom-up dynamic method: 1) compute the optimal aggregated FIB for the sub-tries; 2) based on the results of sub-tries, compute the final optimal aggregated FIB. We proposed the full version of this algorithm in our work of [7]. Fig. 3(c) shows an example.

Both of the two methods can reduce more prefixes than Level-1 FIB Aggregation [4]. Although the second involves more computation overhead, it produces the minimized aggregated FIB (in the case of no packing prefix). Therefore, in this paper, we choose the second one to implement Level-1 NSFIB aggregation algorithm. However, we still believe the first method is another available choice for routers that lack computation capacity, because the first method involves less computational overhead than the second one.



(a) An example FIB (b) First method (c) Second method

Fig. 3. Two methods for Level-1 NSFIB Aggregation. (a) is a NSFIB; (b) is the aggregated FIB produced by the first method; (c) is the aggregated FIB produced by the second method.

B. Level 2-4 NSFIB Aggregation

Analogous to single-nexthop FIB aggregation, NSFIB can be further aggregated by packing some nonexistent prefixes. In this section, we provide the approaches of Level 2-4 NSFIB aggregation algorithms.

In Level-2 single-nexthop FIB aggregation, a prefix covering two sibling prefixes is inserted to aggregate the two children. In Level-2 NSFIB aggregation, we decompose the procedure into two steps: 1) pack a parent prefix for sibling prefixes that share at least one common next hop and set the next hop set of the packed prefix as the union of the two sibling prefixes' next hop sets; 2) aggregate the transformed NSFIB by the second method of Level-1 NSFIB Aggregation.

Fig. 4 shows an example of step 1). The two sibling prefixes are not immediately removed after inserting the parent prefix and the next hop for the inserted prefix is not selected in the first step. This is because of two reasons. **Optimality**: the optimal decision of selecting the next hop for the inserted prefix cannot be made locally. For example, in Fig. 4, if the right child has two descendant prefixes with the next hop of c, a local decision cannot guarantee the optimality. **Stability**: for the stability of the aggregated FIB, we prefer not inserting new prefixes. For example, in Fig. 4, if the right child has one descendant prefix with the next hop of c, the new prefix is not necessary and it will be removed in Step 2).

For Level-3 and Level-4 Single-Nexthop FIB Aggregation, we also provide the corresponding NSFIB Aggregation algorithms. They are decomposed into two steps as Level-2 NSFIB Aggregation for the same reasons of optimality and stability.

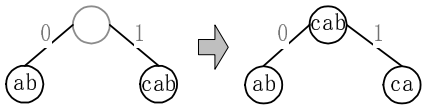


Fig. 4. An example of Step 1) in Level-2 NSFIB Aggregation. Each binary tree represents a sub-trie (or part) of the NSFIB.

C. Optimal Routing Table Construction on NSFIB

Although ORTC algorithm can be directly used to compute the minimized aggregated FIB for a NSFIB, the computation involves too much memory overhead [3]. The space complexity of ORTC algorithm is $O(W|P|)$ (for IPv4, $W = 32$). We make two improvements in NSFIB-ORTC() as suggested in [3]: 1) we extend ORTC from unibit trie to path-compressed trie; 2) we skip the first step of ORTC. The space complexity of NSFIB-ORTC() is $O(|P|)$ while the time complexity is at the same level of ORTC [3].

A path-compressed trie is a trie where no one-way branch node exists. A trie can be transformed into a path-compressed trie by the following principle [13]: 1) remove all the one-way branch nodes; 2) compress the branch information of the removed one-way branch nodes into their first non one-way branch node. To guarantee the search operation can be performed correctly, the branch information must be kept to indicate which bits are bypassed (*bit string*).

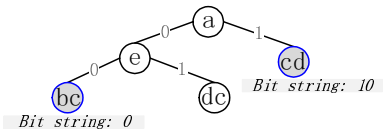


Fig. 5. An example of the path compressed trie.

Before diving into the details of NSFIB-ORTC(), we first introduce some involved definitions. Let n be a node in the path-compressed trie of $\mathcal{F}_{ns}$ (n might be a prefix node or a empty node). Let A_n be the next hop set of the node n ($A_n = \emptyset$ if n is an empty node). Let $prev(n)$ be the set of most prevalent next hops at n. Let $lchild(n)$ ($rchild(n)$) be the left (right) child of n. Let $cand(n)$ be the candidate set of the most prevalent next hops at n.

$$cand(n) = \begin{cases} A_n \cup prev(lch(n)) \cup prev(rch(n)) \setminus \{\delta\}, & A_p \neq \emptyset \\ \{\delta\} \cup prev(lch(n)) \cup prev(rch(n)), & A_n = \emptyset \end{cases}$$

δ represents the next hop that the current empty node inherits from its non-empty ancestor node. Note that $prev(n) \subseteq cand(n)$. Let $bitlen(p)$ be the bit string length of p.

Algorithm 2 NSFIB-ORTC($\mathcal{F}_{ns}$)

```
1: for all n, any node in F_ns (post-order) do
2:    prevalent(n) ← ∅;
3:    for all x ∈ cand(n) do
4:       Ω(x) ← 0, max ← 0;
5:       if x ∈ A_p then
6:          Ω(x) ← Ω(x) + bitlen(lch(n)) + bitlen(rch(n));
7:       if x ∈ prev(lchild(n)) then
8:          Ω(x) ← Ω(x) + 1;
9:       if x ∈ prev(rchild(n)) then
10:         Ω(x) ← Ω(x) + 1;
11:      if Ω(x) > max then
12:         max ← Ω(x);
13:   end for
14:   for all x ∈ cand(n) do
15:      if account(x) = max then
16:      prev(p) ← prev(n) ∪ {x};
17:   end for
18: end for
19: Construct the optimal aggregation by top-down order according
    to the most prevalent next hops at each node;
```

V. Path Stretch Control of NSFIB Aggregation

In this section, we provide another dimension for routers to make the trade-off: path stretch *VS.* aggregation performance. In order to shrink more entries, NSFIB-based aggregation algorithms might select non-optimal next hops. Routers with larger line card capacity might have the demand to filter some *worse* (selectable) next hops to alleviate the path stretch even though this would decrease the aggregation performance.

The intra-domain path stretch is determined by all the selected next hops along the path. However, in order to maintain the router-level incremental deployability of NSFIB-based aggregation, a local approach to filter *worse* next hops is required. Let R_n be an SPO next hop from R_s to R_d. We define the one-step increase amplitude of R_n as $cost(R_s, R_n) + dist(R_n, R_d) - dist(R_s, R_d)$. Intuitively, the next hops with greater one-step increase amplitudes are *worse*. Routers can filter some *worse* next hops by setting the maximum number of next hops. For a certain destination, the next hops exceeding the number will be filtered according to the one-step increase amplitude. Routers can make a trade-off between the aggregation performance and path stretch by setting a proper value for the maximum number of next hops.

VI. Simulation Results

In order to demonstrate the performance of FIB aggregation algorithms, including single-nexthop FIB aggregation and NSFIB aggregation, we use diverse topologies generated by BRITE [10]: The topology sizes range from 200 to 1200; the average degrees range from 4 to 28. We use the RIB from Route View to construct the routing system. The RIB (obtained in Feb. 2012) has 410,431 prefixes.

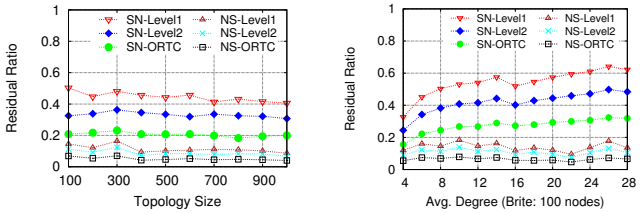
A. Residual Ratio

We first show the results of residual ratio: the aggregated FIB size divided the original FIB size. Smaller residual ratio means better shrinking performance. We select six representative algorithms: single-nexthop Level-1 FIB Aggregation (SN-LRVEL1), SN-LEVEL2, SN-ORTC, NSFIB-Level1 (NS-Level1), NS-Level2 and NS-ORTC.

As Fig. 6(a) shows, the residual ratios of SN-LEVEL1, SN-LEVEL2 and SN-ORTC are respectively about 46%, 35% and 22% in the topologies with the degree of 8, compared with 11%, 9% and 6% of NS-Level1, NS-Level2 and NS-ORTC. All the NSFIB-based algorithms perform much better than the corresponding single-nexthop algorithms. Fig. 6(b) shows the relation between the residual ratio and topology degree. Note that the shrinking perform of single-nexthop algorithms degrades as the topology degree increases, while NSFIB-based algorithms are not affected by the topology degree.

B. Path Stretch Control

Now we show the results when setting an upper limit for the selectable next hops. First, we show the relation between the residual ratio and the upper limit. The residual ratio increases (Fig. 7) and the path stretch ratio decreases (Fig. 8) as the upper limit decreases from X (no limit) to 1. When the upper limit is set to be 1, NSFIB-based algorithms degrade into single-nexthop FIB algorithms and the path stretch is zero. This proves that our path control method is an efficient way to make a trade-off between the shrinking performance and the path stretch. A router can set the upper limit according to its memory capacity.



(a) Topology degree: 8 (b) Topology size: 100

Fig. 6. The residual ratios of different aggregation algorithms on BRITE topologies. a) is the relation between the residual ratio and topology size; b) is the relation between the residual ratio and topology degree.

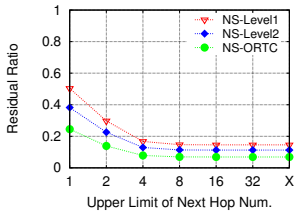


Fig. 7. The relation between the upper limit of selectable next hops and the residual ratios of NSFIB-based algorithms.

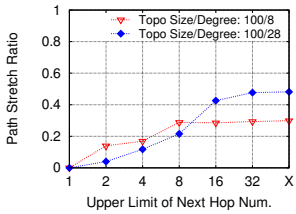


Fig. 8. The relation between the upper limit of selectable nexthop number and the path stretch of NSFIB-Level1 algorithm.

References

[1] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," RFC 4984, Sep. 2007.

[2] T. R. V. Project, http://www.routeviews.org.

[3] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, "Constructing optimal IP routing tables," in *Proc. IEEE INFOCOM*, April 1999.

[4] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the aggregatability of router forwarding tables," in *Proc. IEEE INFOCOM*, San Diego, CA, March 2010.

[5] Y. Liu, X. Zhao, K. Nam, L. Wang, and B. Zhang, "Incremental forwarding table aggregation," in *Proc. IEEE GLOBECOM*, Miami, Florida, USA, November 2010.

[6] Z. A. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis, "Practical and Near-Optimal FIB aggregation with SMALTA," in *Proc. CoNEXT*, Tokyo, Japan, December 2011.

[7] Q. Li, D. Wang, M. Xu, and J. Yang, "On the scalability of router forwarding tables: Nexthop-selectable fib aggregation," in *Proc. IEEE INFOCOM*, April 2011.

[8] T. C. Education and R. N. (CERNET), http://www.edu.cn/english.

[9] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proc. IEEE IMW*, Marseille, France, Nov. 2002.

[10] BRITE, http://www.cs.bu.edu/brite.

[11] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless inter-domain routing (CIDR): an address assignment and aggregation strategy," RFC 1519, Sep. 1993.

[12] P. Narvaez, K. Siu, and H. Tzeng, "New dynamic SPT algorithm based on a ball-and-string model," *Transactions on Networking*, vol. 9, pp. 706–718, 2001.

[13] M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, no. 2, pp. 8–23, 2001.