NSFIB Construction & Aggregation with Next Hop of Strict Partial Order

Qing Li, Mingwei Xu, Meng Chen
Department of Computer Science & Technology, Tsinghua University
{liqing, xmw, chenmeng}@csnet1.cs.tsinghua.edu.cn

***Abstract*—The Internet global routing tables have been expanding at a dramatic and increasing rate. Although the latest high-performance routers provide enough memory capacities, the Internet Service Providers cannot afford to upgrade their routers at the pace of routing table growth. In this paper, we propose the next hop of strict partial order (SPO next hop) and a corresponding algorithm to construct the Nexthop-Selectable FIB (NSFIB). In NSFIB, each prefix has multiple next hops. Based on the constructed NSFIB, we design different levels of aggregation algorithms, from which a router can select a proper one according to its memory capacity. Besides, we provide a method of setting an upper limit to filter SPO next hops leading to higher path length. A router can make a trade-off between aggregated FIB size and path stretch by adjusting the upper limit. We also introduce routing protection by the pre-computed SPO 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; and SPO next hops protect about 70% (topology-related) failure-affected packets.**

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. Due to multi-homing, traffic engineering, policy routing, *etc.*, unaggregatable address fragments from edge networks are continuously pouring into the backbone of Internet. By July 2012, the routing table size has reached 433,653 (from 15,100 of 1994) [2]. Although the latest high-performance routers provide enough memory capacities, the Internet Service Providers (ISPs) cannot afford to upgrade their routers at the pace of routing table growth [3, 4]. Therefore, effective solutions have yet to be identified.

One approach of handling this Internet routing scalability problem is to separate the edge network addresses from the backbone. Numbers of long-term solutions by this approach have been proposed, including Shim6 [5], ILNP [6], LISP [7], APT [8], MILSA [9], SIX/ONE [10], *etc*. Although these long-term solutions can control the routing table size, they change the current Internet routing protocols or even involve a brand new routing protocol. A long time is required to deploy these solutions, according to the experience of IPv6. Besides, there are also some non-IP solutions, such as compact routing [11] and geographic routing [12]. Nevertheless, these solutions do not target directly on the current Internet.

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 [13–16] cannot provide a satisfactory aggregation performance. Besides, they have a common problem: the performance degrades as the degree of the network topology increases [17].

We proposed Nexthop-Selectable FIB (NSFIB) aggregation in [17]. 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, NSFIB-based aggregation can achieve a better performance in FIB shrinking. 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 [13–16] 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 [13–16] on NSFIB. We select two typical ones to demonstrate the algorithms: 1) according to the four levels of single-nexthop aggregation algorithms in [14], 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 [13] 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.

We also introduce the architecture of routing protection. As multiple SPO next hops are computed during NSFIB construction, they can be used as backup next hops to protect packets during network failures. The packet that is affected by a network failure will be dropped before the routing protocol converges (*i.e.*, the route computation finishes). In our scheme, another SPO next hop can be used when the currently selected next hop fails, which will decrease the packet dropping ratio during network failures.

Finally, we demonstrate the performance of NSFIB aggregation algorithms by comprehensive simulations on China Education and Research Network (CERNET) [18], Rocketfuel [19] topologies and BRITE-generated topologies [20]. 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. 3) The SPO next hops can avoid about 70% packet loss during the failure. In denser networks, the protection ratio can even reach 95%.

II. Background and Related Works

It is widely accepted that the Internet routing system is facing severe routing scalability problem [1]. To solve this problem, numbers of solutions [6–12] have been proposed to change or upgrade the architecture of Internet. However, these long-term solutions involve protocol change or nontrivial configuration overhead, thus they demand a long deploying time, like IPv6. Therefore, several FIB aggregation schemes, which are router-level incrementally deployable, have been proposed to handle the immediate requirement (*i.e.*, shrinking the FIB) of ISPs. In this section, we provide a brief review of these schemes. Before introducing these FIB aggregation schemes, we first make some explanations of the notations.

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.

Fig. 1(a) show a toy example of trie corresponding to the FIB of $\{< *, a >, < 00*, b >, < 10*, b >, < 11*, c >\}$. Note that some nodes in the radix tree correspond to real prefixes in the FIB and others do not. We call the former and latter as *non-empty* nodes and *empty* nodes, respectively.

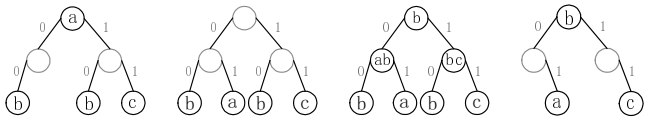
A. Single-Nexthop FIB Shrinking Schemes

In this paper, we refer to the general FIB where each prefix has one single next hop (*i.e.*, the optimal one to the destination) 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 [14]. The principle of General FIB Aggregation has been widely used for routing table aggregation since Classless Inter-Domain Routing (CIDR) [21] 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.

General FIB Aggregation is router-level incrementally deployable. However, it can achieve only 30-40% reduction in FIB size [17], which is not satisfactory to ISPs.

2) Optimal Routing Table Constructor and Successors: To further shrink the single-nexthop FIB, Draves, *et al.*, proposed *Optimal Routing Table Constructor* (ORTC) [13]. 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. The normalized FIB is equal with the original one in forwarding behavior. **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 and generate the aggregated FIB. 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

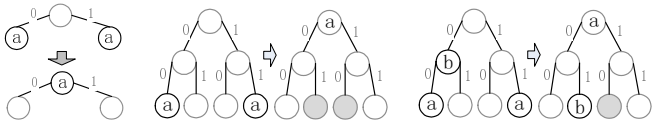
Fig. 1. The steps of Optimal Routing Table Constructor (ORTC).

ORTC produces the minimized FIB with the equivalent forwarding behavior with the original one. However, ORTC itself does not provide an efficient online algorithm and it involves nontrivial memory overhead [16]. To complement ORTC, Incremental Forwarding Table Aggregation (IFTA) [15] and Smalta [16] have been proposed. These two successors provide

non-optimal efficient online algorithms for ORTC. After some updates, the aggregated FIB drifts away from the optimal one. ORTC is periodically called to restore the optimality.

3) Level 1-4 Aggregation: Level 1-4 aggregations [14] were proposed to provide different levels of FIB aggregation (better aggregation & higher complexity from 1 to 4). ISPs with different requirements can select proper levels to deploy. Level 1 aggregation is General FIB Aggregation.

Different from ORTC, Level 2-4 aggregations [14] 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 [14]. 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 [17], we proposed Nexthop-Selectable FIB (NSFIB) aggregation to further compress FIB by constructing multiple selectable next hops for each prefix. Fig. 3 shows how NSFIB works in a router. A NSFIB is a set of entries, like a FIB, except that in each entry, the 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 by the covering one. For example, $< 101*, \{a, b\} >$ can be aggregated by $< 10*, \{b\} >$. As the possibility of two prefixes sharing one common next hop is higher than the possibility of two prefixes having the same optimal next hop, NSFIB aggregation aggregates more entries than single-nexthop FIB aggregation. We assumed that NSFIB is constructed by the DownStream Condition (DCS) [22]. We developed the NSFIB aggregation algorithm of polynomial time.

However, as a first-step work, NSFIB aggregation [17] has several problems to be solved before practical deployment:

- An efficient NSFIB construction algorithm. Network failures are inevitable and the failures will trigger NSFIB reconstruction, therefore, an efficient NSFIB construction algorithm is required.
- The algorithms of applying different single-nexthop FIB aggregation methods [13–16] on the NSFIB.
- An approach to control the path stretch caused by NSFIB aggregation, which might use non-optimal next hops.

III. Next Hop of Strict Partial Order

In this section, we first propose an improved NSFIB construction approach of SPO next hop; we then provide an efficient construction algorithm according to the SPO next hop, which has the same time complexity as the algorithm of shortest path first (SPF); finally, we prove that based on the construction approach of SPO next hop, the performance of NSFIB aggregation increases as the density (average degree) of the network increases.

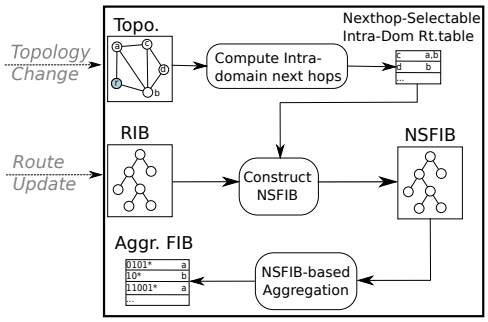


Fig. 3. NSFIB aggregation in a router.

A. Next Hop of Strict Partial Order

Open Shortest Path First (OSPF) is widely used as the intra-domain routing protocol in the Internet. Loop Free Alternates (LFA) [22] is an intra-domain scheme for failure protection, which is based on OSPF and thus well compatible with the current Internet. In LFA, there are several conditions for backup next hops (alternates) selection, including DownStream Condition (DSC). In [17], we directly used DownStream Condition to construct NSFIB. However, we found that this method could be improved for better performance. Therefore, we propose the next hop of strict partial order (SPO next hop).

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 (*i.e.*, irreflexive, antisymmetric and transitive).

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 $R_n \prec_{R_d} R_s$.*

It is straightforward that the optimal next hop is a SPO next hop. Because the optimal next hop is closer to the destination than the source node.

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

Proof: Actually, a strict partial order corresponds directly to a directed acyclic graph, therefore, forwarding by the SPO next hop guarantees no routing loop. Here we provide a detailed proof for this theorem by contradiction. 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. The aggregation will make no impact on other ASes, because the egress router, which connects to the next AS, will not be changed by the aggregation.

Fig. 4 shows an example of SPO next hops. As $dist(R_s, R_d) = 10$ and $dist(R_y, R_d) = 9$, $R_y \prec_{R_d} R_s$, thus R_y is a SPO next hop from R_s to R_d. As $dist(R_s, R_d) = dist(R_x, R_d) = 10$ and $ID(R_x) < ID(R_s)$, $R_x \prec_{R_d} R_s$, thus R_x is also a SPO next hop from R_s to R_d. R_w and R_z are not SPO next hops from R_s to R_d according to Definition 1. Assume R_d is the egress router of the prefix $1001*$, the corresponding entry for $1001*$ in the NSFIB of R_s is $\langle 1001*, \{R_x, R_y\}\rangle$.

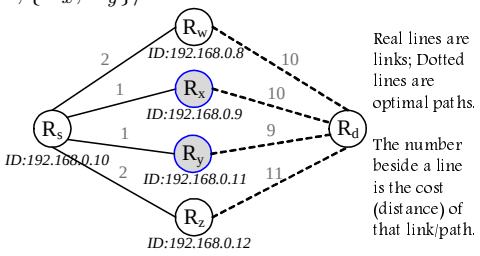


Fig. 4. An example of SPO next hops.

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) [23] 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. The computation complexity is the same as one time of SPF.

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**

C. Qualitative Analysis of NSFIB-Based Aggregation

Assume all the neighbors have the same opportunity to become the optimal next hop (or selectable next hop). If the router has K neighbors, the possibility of any two prefixes having the same next hop in the single-nexthop FIB is $\frac{1}{K}$. In single-nexthop FIB aggregation schemes [13, 14, 16], more neighbors mean less aggregation achievement. Therefore, as the density (average degree) of the topology increases, the achievement of the single-nexthop FIB aggregation decreases. This limits the practical contribution of these single-nexthop FIB aggregation schemes because large backbone networks with higher densities are actually more urgent to shrink FIBs.

Given an intra-domain destination R_d, for any pair of neighbors R_a and R_b, one and only one of the following propositions is true: 1) R_a is a SPO next hop from R_b to R_d; 2) R_b is a SPO next hop from R_a to R_d. A straightforward proof can be established according to the definition of SPO next hop. Therefore, the expected number of SPO next hops for any prefix is half of the number of neighbors. If the router has K neighbors, each prefix in its FIB has $\frac{K}{2}$ SPO next hops in average. As we are focusing on a qualitative theoretical analysis, we directly assume that each prefix has $\frac{K}{2}$ SPO next hops randomly among the router's neighbors. In this case, the possibility of any two prefixes sharing at least one common SPO next hop is $\Gamma = 1 - \frac{\frac{K}{2}(\frac{K}{2}-1)...1}{K(K-1)...(\frac{K}{2}+1)}$.

Note that $\Gamma > \frac{1}{K}$ when $K > 1$. Thus, the aggregation achievement of our NSFIB is better than single-nexthop FIB. More significantly, Γ increases while K increases, which means that as the density (average degree) of the topology increases, the aggregation achievement of our NSFIB also increases. Although the increasing can be almost ignored when $K > 6$, the performance of NSFIB-based aggregation, at least, does not degrade as the topology degree increases, which is a great improvement over single-nexthop FIB aggregation.

IV. NSFIB-Based Aggregation Algorithms

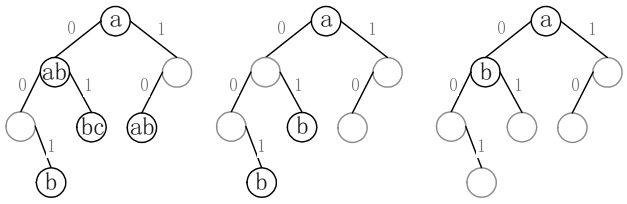
Let a FIB be a set $\mathcal{F} = \{\langle p, a_p\rangle | p \in P\}$ where P is the set of prefixes in the FIB and a_p is the next hop for prefix p. Let a Nexthop-Selectable FIB (NSFIB) be a set $\mathcal{F}_{ns} = \{\langle p, A_p\rangle | p \in P_{ns}\}$ where P_{ns} is the prefix set and A_p is 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 [14]). *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. 5(b) shows an aggregated FIB by this algorithm with the NSFIB of Fig. 5(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 [17]. Fig. 5(c) shows an aggregated FIB by this algorithm with the NSFIB of Fig. 5(a) as the input.



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

Fig. 5. 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.

Both of the two methods can reduce more prefixes than Level-1 FIB Aggregation [14]. 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.

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. 6 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. 6, 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. 6, 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. 7 shows the example.

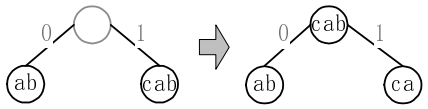
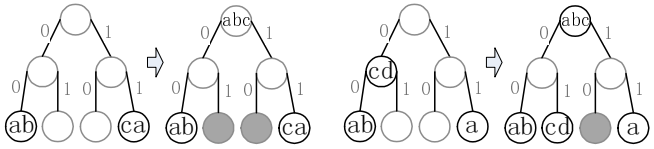


Fig. 6. An example of Step 1) in Level-2 NSFIB Aggregation. Each binary tree represents a sub-trie (or part) of the NSFIB.

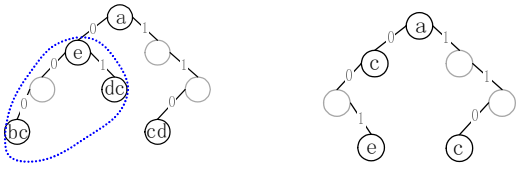


(a) Level 3: Step 1) (b) Level 4: Step 1)

Fig. 7. Step 1) in Level-3 and Level-4 NSFIB Aggregation.

C. Optimal Routing Table Construction on NSFIB

1) Problem Formulation: A FIB $\mathcal{F}$ is a feasible aggregation for a NSFIB $\mathcal{F}_{ns}$ iff the following condition is satisfied: for any IP address, $\langle p, a_p \rangle$ and $\langle p_{ns}, A_{p_{ns}} \rangle$ are respectively its matched entries in $\mathcal{F}$ and $\mathcal{F}_{ns}$, we have $a_p \in A_{p_{ns}}$, *i.e.*, the longest matched next hop in $\mathcal{F}$ is a selectable next hop in $\mathcal{F}_{ns}$. The **NSFIB-ORTC Problem** is: given a NSFIB $\mathcal{F}_{ns}$, find a minimized feasible aggregation $\mathcal{F}$ for $\mathcal{F}_{ns}$.

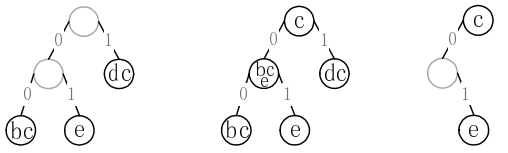


(a) A NSFIB (b) An optimal aggregation

Fig. 8. (a) shows a NSFIB where the first letters in circles are the optimal next hops for the corresponding prefixes and the other letters are non-optimal selectable next hops; (b) shows an optimal aggregation for the NSFIB.

In Fig. 8, we show an example. Fig. 8(a) is a NSFIB including the five entries. The first letters in the circle are the optimal next hops; the other letters are non-optimal selectable next hops. Single-nexthop FIB aggregation algorithms (including ORTC) and Level 1-4 NSFIB aggregation algorithms cannot shrink the NSFIB. However, NSFIB-ORTC can shrink the NSFIB into four entries, as Fig. 8(b) shows.

2) Algorithm Design: The algorithm of ORTC [13] can be directly applied to solve the NSFIB-ORTC problem. We show the steps of ORTC with the sub-trie in the dotted line of Fig. 8(a) as the input. Fig. 9(a) is the normalized NSFIB after Step 1. Step 2 computes the most prevalent next hops at each level by bottom-up order, as shown in Fig. 9(b). Step 3 removes the prefixes that can be aggregated and determines the next hop for each prefix, as shown in Fig. 9(c). We can see that the result is the same with the corresponding part of Fig. 8(b).



(a) Step 1 (b) Step 2 (c) Step 3

Fig. 9. The three steps of ORTC with the input of the sub-trie (a NSFIB) in the dotted line of Fig. 8(a).

Although ORTC algorithm computes the minimized aggregated FIB for a NSFIB, the computation involves too much memory overhead [13]. The space complexity of ORTC algorithm is $O(W|P|)$ (for IPv4, $W = 32$; for IPv6, $W = 128$). To solve the problem of NSFIB ORTC, we make two improvements in our algorithm of NSFIB-ORTC() as suggested in [13]: 1) we extend the ORTC algorithm 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 [13].

In a binary trie, there may exist some empty node with only one child (see the nodes of $00*$, $1*$ and $11*$ in Fig. 8(a)). Even though no branching decision is made at these nodes, they need to be inspected for prefix searching, which may cause an unnecessary longer search time. Besides, these one-child empty (*one-way branch*) nodes consume additional memory. Therefore, we employ path-compressed trie in our algorithm.

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 [24]: 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. Fig. 10 shows an example of path-compressed trie corresponding to the binary trie in Fig. 8(a). To guarantee the search operation can be performed correctly, the branch information must be kept to indicate which bits are bypassed (*bit string*), as shown in Fig. 10.

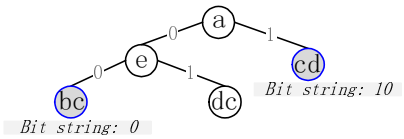


Fig. 10. The corresponding path compressed trie of the NSFIB in Fig. 8(a).

Given the path-compressed trie of $\mathcal{F}_{ns}$, NSFIB-ORTC() computes the optimal aggregation by two steps. Lines 1 to 23 are the first step, which computes the most prevalent next hops at each node of the path-compressed trie (by bottom-up order). Based on the first step, the second step determines the next hop for each prefix, removes the prefixes that can be aggregated and splits the prefix if necessary (by top-down order).

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

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

1: **for all** n, any node in $\mathcal{F}_{ns}$ (post-order) **do**
2: $prevalent(n) \leftarrow \emptyset$;
3: **for all** $x \in cand(n)$ **do**
4: $\Omega(x) \leftarrow 0, max \leftarrow 0$;
5: **if** $x \in A_p$ **then**
6: $\Omega(x) \leftarrow \Omega(x) + bitlen(lch(n)) + bitlen(rch(n))$;
7: **end if**
8: **if** $x \in prev(lchild(n))$ **then**
9: $\Omega(x) \leftarrow \Omega(x) + 1$;
10: **end if**
11: **if** $x \in prev(rchild(n))$ **then**
12: $\Omega(x) \leftarrow \Omega(x) + 1$;
13: **end if**
14: **if** $\Omega(x) > max$ **then**
15: $max \leftarrow \Omega(x)$;
16: **end if**
17: **end for**
18: **for all** $x \in cand(n)$ **do**
19: **if** $account(x) = max$ **then**
20: $prev(p) \leftarrow prev(n) \cup \{x\}$;
21: **end if**
22: **end for**
23: **end for**
24: Construct the optimal aggregation by top-down order according to the most prevalent next hops at each node;

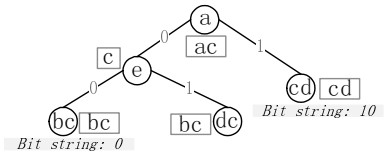
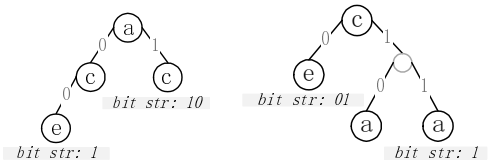


Fig. 11. The most prevalent next hops at each node of the path-compressed trie in Fig. 10. The letters in the rectangle are the most prevalent next hops.



(a) Optimal Aggregation 1 (b) Optimal Aggregation 2

Fig. 12. The two optimal aggregations of Fig. 10. Optimal Aggregation 1 is the path-compressed form of Fig. 8(b).

($A_n = \emptyset$ 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.

Based on the above definitions, we can see that lines 1-23 in Algorithm 2 compute the most prevalent next hops at each node of $\mathcal{F}_{ns}$. Fig. 11 shows an example with the path-compressed trie of Fig.10 as the input. According to the computed prevalent next hops, we construct two optimal aggregations with $*$ selecting a or c respectively, as shown in Fig. 12. In our algorithm, we choose the first one to reduce the dynamics of the aggregated FIB according to the following principle: for the same prevalent next hops, the one the prefix originally has is prior to the others.

V. Path Stretch Control of NSFIB Aggregation

In the real network, routers should choose the proper aggregation algorithms according to the capacity of their line cards. In the previous section, we provide several levels of NSFIB-based algorithms with different computation complexities. Routers with larger line card capacity should choose the algorithm with lower computation complexity (lower aggregation performance). 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 caused 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. Routing Protection by SPO Next Hops

Network failures are inevitable in the Internet. Failure-affected packets are dropped before the routing protocol converges on the new topology [25]. In order to avoid packet dropping, routing protection has been proposed to reroute the failure-affected packets by a pre-computed next hop (path) [22, 26, 27]. In our scheme, the pre-computed SPO next hops can also be used for routing protection.

Currently, a software trie is used in the router to store the BGP routing table, as shown in Fig. 3. In our scheme, the NSFIB (with SPO next hops) before aggregation is also stored in the same trie, with some extending attributes for each node. The aggregated FIB is stored in the line card as customary. A router R_s forwards a packet as follows:

1) Look up the destination in the aggregated FIB (stored in the line card) by longest prefix matching principle and get the next hop R_n for the packet;
2) If R_s detects the failure of R_n, go to 3); else, forward the packet to R_n and finish the processing of the packet;

3) Look up the destination in the NSFIB trie and get the SPO next hops, select one to forward the packet.

VII. Simulation Results

A. Simulation Setup

In order to demonstrate the performance of FIB aggregation algorithms, including single-nexthop FIB aggregation and NS-FIB aggregation, we obtain three sets of topologies.

- The real topology from China Education and Research Network (CERNET) [18]: CERNET is one of the most significant networks in China, which connects more than 1,500 education or research institutions.
- The six real topologies from Rocketfuel [19] (Table I).
- Diverse topologies generated by BRITE [20]: The topology sizes range from 200 to 1200; the average degrees range from 4 to 28, as shown in Table II.

For the topologies from Rocketfuel and BRITE, we use the RIB from Route View to construct the routing system. The RIB (obtained in Feb. 2012) has 410,431 prefixes.

TABLE I
The Characteristics of Rocketfuel AS Topologies

AS Number	Name	#Routers	#Links	Average Degree
1221	Telstra (au)	104	151	2.90
1239	Sprint (us)	315	972	6.17
1755	Ebone (eu)	87	161	3.70
3257	Tiscali (eu)	161	328	4.07
3967	Exodus (us)	79	147	3.72
6461	Abovenet (us)	128	372	5.81

TABLE II
Parameters of BRITE Topologies

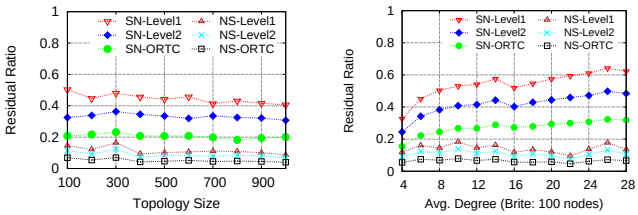
Mode	Model	HS	LS	Nodes Num
Router Only	Waxman	1000	100	50-500
links/new node	α / β	NP	Growth Type	
2-14	0.15 / 0.2	Random	Incremental	

B. 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. 13(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. 13(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. This is consistent with our analysis in Section III-C.

As Fig. 14 shows, the results on the six Rocketfuel topologies are consistent with the results of BRITE topologies. The



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

Fig. 13. 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.

single-nexthop FIB aggregation algorithms (SN-Level1, SN-Level2 and SN-ORTC) perform better on Rocketfuel topologies than on BRITE topologies. This is because that the Rocketfuel topologies have smaller topology degrees. The results of CERNET topology show the same property. Due to the limited space, we ignore the detailed results in CERNET.

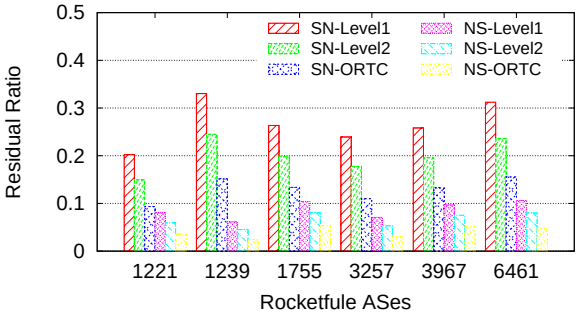
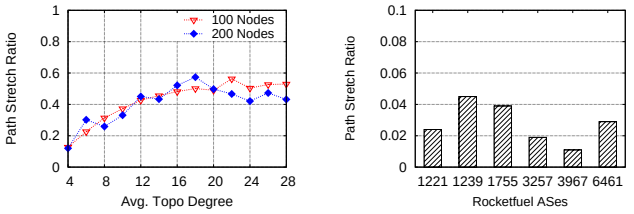


Fig. 14. The residual ratios of different aggregation algorithms on Rocketfuel topologies. (a) shows the results of Level-1 FIB/NSFIB aggregation; (b) shows the results of ORTC and NSFIB ORTC.

C. Path Stretch Control

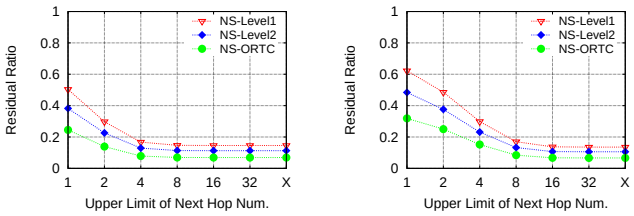
As not all the SPO next hops are optimal, NSFIB-based aggregation algorithms introduces intra-domain path stretch. We select Level-1 NSFIB aggregation to show the path stretch ratios and the function of our path stretch control method. Generally, not all the routers require FIB aggregation. We assume that 60% of routers select the NSFIB aggregation. First, we show the path stretch without controlling in Fig. 15. The path stretch ratios range from 10% to 55% for Brite topologies. The higher topology degree means more SPO next hops, *i.e.*, the optimal next hop is less likely to be selected in the aggregated FIB. Therefore, the path stretch ratio increases as the degree increases. However, the increasing stops after the degree of 16. This can be explained as follows. In our algorithm, if two SPO next hops both lead to the minimized aggregated FIB, the one with the shorter path will be selected. Therefore, the increasing of path stretch stops when the number of neighbors exceeds a certain threshold. As the degrees of Rocketfuel topologies are all less than 7, the path stretch ratios are much lower ($< 5\%$).

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. 16) and the path stretch ratio decreases (Fig. 17) 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) Stretch of Brite topo. (b) Stretch of Rocketfuel topo.

Fig. 15. The path stretch ratios of Level-1 NSFIB Aggregation. (a) shows the results on BRITE topologies; (b) shows the results on Rocketfuel topologies.



(a) Topo. Degree: 8 (b) Topo. Degree: 28

Fig. 16. The relation bwtween the upper limit of selectable next hops and the residual ratios of NSFIB-based algorithms (Brite topologies of 100 nodes). (a) The topology degree is 8; (b) The topology degree is 28.

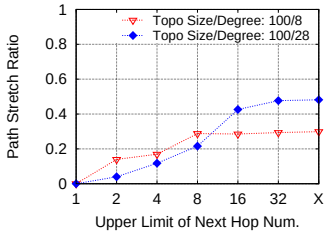


Fig. 17. The relation between the upper limit of selectable nexthop number and the path stretch of NSFIB-Level1 algorithm.

D. Protection Ratio

As discussed in Section VI, the pre-computed SPO next hops can be used to re-route the failure-affected packets when failure occurs in the network. We randomly generate 10,000 failures in Brite topologies. According to [28], we set 70% of them as single-link failures, 15% of them as single-node failures and the remained as double-link failures. The results show that SPO next hops provide protection coverage from 50% (low-degree topologies) to 95% (high-degree topologies). Due to the limited space, we ignore the detailed results.

VIII. Conclusion

In this paper, we made the following contributions. First, we designed an efficient algorithm to construct the NSFIB and proved the performance of NSFIB-based aggregation would not degrade as the topology degree increases, which is a great improvement over single-nexthop FIB aggregation. The comprehensive simulation results further confirm our analysis. Then, we designed several NSFIB-based algorithms, *e.g.*, NS-FIB Level 1-4 aggregation and NSFIB-ORTC. The simulation results show that NSFIB-based algorithms can shrink the FIB size to 5% to 20%, compared with 20% to 65% of single-nexthop FIB aggregation algorithms. After that, we provided an approach to control the path stretch caused by NSFIB-based algorithms. Besides, we introduced routing protection by the pre-computed SPO next hops. As multiple SPO next hops are computed during NSFIB construction, they can be used as backup next hops to protect packets during network failures. The packet that is affected by a network failure will be dropped before the routing protocol converges (*i.e.*, finish the route computation). In our scheme, another SPO next hop can be used when the currently selected next hop fails, which will decrease the packet dropping ratio during network failures. The simulation results show that SPO next hops can provide about 70% protection coverage.

In all, no FIB aggregation algorithm performs better in all dimensions. We were not aiming to provide an "almighty" aggregation algorithm but provide flexible choices for different scenarios. A router should select the proper aggregation algorithm and set a proper value for the upper limit of next hop number according to its own memory capacity. According to the complexity and FIB shrinking performance, routers with higher memory capacity should select single-nexthop FIB Level 1-2 aggregation; routers with lower memory capacity should select NSFIB Level 1-2 aggregation. However, if the router has the requirement of routing protection, NSFIB Level 1-2 aggregation should be preferred. Nevertheless, ORTC (single-nexthop or NSFIB) should be the last choice because of its high computation and memory overhead.

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] X. Zhao, D. J. Pacella, and J. Schiller, "Routing scalability: An operator's view," *JSAC*, vol. 28, no. 8, pp. 1262–1270, 2010.

[4] V. Khare, D. Jen, X. Zhao, Y. Liu, D. Massey, L. Wang, B. Zhang, and L. Zhang, "Evolution towards global routing scalability," *JSAC*, vol. 28, no. 8, pp. 1363–1375, 2010.

[5] E. Nordmark and M. Bagnulo, "Shim6: Level 3 multihoming shim protocol for IPv6," Internet RFC 5533, Jun. 2009.

[6] R. Atkinson, S. Bhatti, and S. Hailes, "ILNP: mobility, multi-homing, localised addressing and security through naming," *Telecommunication Systems*, vol. 42, no. 3, pp. 273–291, 2009.

[7] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/id separation protocol (LISP)," Internet Draft, draft-ietf-lisp-07, Apr. 2010.

[8] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang, and L. Zhang, "Towards a new internet routing architecture: Arguments for separating edges from transit core," in *Proc. HotNets*, Calgary, Alberta, Oct. 2008.

[9] J. Pan, R. Jain, S. Paul, and C. So-in, "MILSA: a new evolutionary architecture for scalability, mobility, and multihoming in the future internet," *JSAC*, vol. 28, no. 8, pp. 1344–1362, 2010.

[10] C. Vogt, "Six/one router: A scalable and backwards compatible solution for provider-independent addressing," in *Proc. MobiArch*, Seattle, WA, Aug. 2008.

[11] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: Routing on flat labels," in *Proc. Sigcomm*, Pisa, Italy, Sep. 2006.

[12] R. Oliveira, M. Lad, B. Zhang, and L. Zhang, "Geographically informed Inter-Domain routing," in *Proc. ICNP*, Beijing, China, Oct. 2007.

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

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

[22] A. Atlas and A. Zinin, "Basic specification for ip fast reroute: Loop-free alternates," RFC 5286, Sep. 2008.

[23] 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.

[24] 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.

[25] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second igp convergence in large IP networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, 2005.

[26] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using not-via addresses," IETF Draft, draft-ietf-rtgwg-ipfrr-notvia-addresses-05, Mar. 2010.

[27] S. Bryant, C. Filsfils, S. Previdi, and M. Shands, "IP fast reroute using tunnels," IETF Draft, draft-bryant-ipfrr-tunnels-03, Sep. 2007.

[28] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proceedings of INFOCOM*, March 2004.