SRUF: Low-Latency Path Routing with SRv6 Underlay Federation in Wide Area Network

Bangbang Ren[1], Deke Guo[1,2*], Guoming Tang[2], Weijun Wang[3], Lailong Luo[1], Xiaoming Fu[3]
National University of Defense Technology[1], Peng Cheng Laboratory[2],University of Goettingen[3]

***Abstract*—Existing Internet routing protocols much focus on providing interconnection service for independent autonomous systems (ASes) rather than end-to-end low latency transmission. Nowadays, a growing number of applications and platforms have high requirements for low latency. However, developing new routing protocols in the wide area network that provides low latency routing service is very challenging, and remains an open problem due to the obstacles of compatibility, feasibility, scalability and efficiency. On the other hand, the ignorance of latency performance results in triangle inequality violations (TIV). In this paper, we leverage TIV and a new routing technology, SRv6, to build a new distribute enhancement routing protocol, SRv6 underlay federation (SRUF), which aims to provide low-latency routing services in network core. We design a novel method to find alternative paths with lower latency between any pair of ASes in SRUF. This method can achieve high scalability as it incurs only $O(n)$ bandwidth overhead in each member of SRUF. SRv6 is then employed to steer the flows along the selected indirect low-latency paths, with keeping compatibility to legacy routing systems. The experimental results with real-world datasets demonstrate that SRUF can effectively reduce the average end-to-end delay by $5.4\% \sim 58.9\%$.**

I. Introduction

Nowadays, many Internet applications have strict requirements for end-to-end latency, e.g., high-frequency e-commerce trading, high-resolution video conference and so on [1]. As the data delivery latency affects users' experiences as well as providers' profits, it has become a critical concern for service providers nowadays [2]. Some large cloud providers like Microsoft and Google leveraged software defined network (SDN) to provide low-latency service in their private networks [3], [4]. These solutions require that the traffic is steered only in the private network which is managed by a central controller. Unfortunately, this requirement is hard to be satisfied in the global Internet. In fact, a cloud provider's private network may not cover its all datacenters. Sometimes, the applications may even be deployed on different providers' clouds considering the tradeoff between price and performance. In consequence, flows from different datacenters could traverse the Internet that is composed of a large number of autonomous systems (ASes). As shown in Fig. 1, there are three datacenters in Shanghai, Guangzhou and Moscow, respectively. The paths between any two datacenters may traverse multiple ASes.

To find routing paths in the wide area network efficiently, two types of routing protocols are necessary. One is the interior gateway protocol e.g., OSPF, which is used to find routing paths inside an AS. The other one is the border gateway protocol (BGP) which is a standardized exterior gateway protocol and used to find routing paths between ASes. These routing protocols work together to build routing paths between any pair of nodes in the wide area network (WAN). However, these paths only ensure connectivity without achieving much in guaranteeing any quality of service. As prior studies [5], [6] have demonstrated, none of the criteria BGP uses for picking up paths directly has an explicit relationship with performance, e.g., preferring paths with fewer AS-level hops and peering over transit, and performing hot potato routing, *etc*. On the other hand, any modifications or optimizations of BGP are hard to implement since BGP is widely adopted in the Internet and deployed in a large amount of devices. Therefore, there is a pressing need to develop new methods that are capable of reducing end-to-end delay and compatible with the existing routing paradigms.

A recent public cloud interconnection test has shown that the default direct paths to connect two datacenters sometimes are worse than indirect paths [7]. Table I summarizes part of the measurement results. The end-to-end delay between Shanghai and Moscow is 262.1ms, while, its packet loss rate could be as high as 25.24% due to the congested links in the

This work is done during B. Ren's visiting time in University of Goettingen. Corresponding author: Deke Guo {dekeguo@nudt.edu.cn}

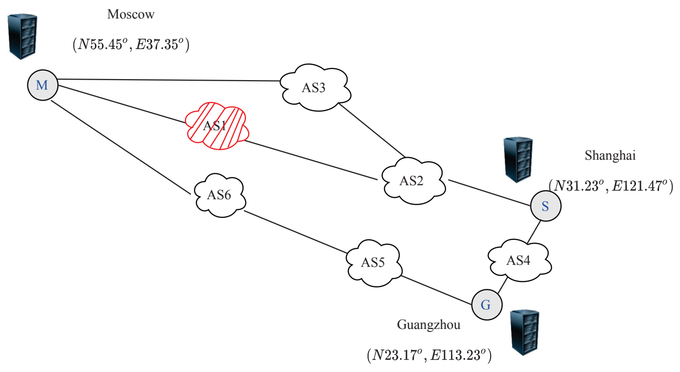


Fig. 1. An example to show inter-domain routing where AS1 is congested, leading to the triangle inequality violation, i.e, $t_{S\rightarrow M} > t_{S\rightarrow G} + t_{G\rightarrow M}$.

TABLE I
THE RESULTS OF LINK MEASUREMENT BETWEEN ANY TWO DATACENTERS INCLUDING END-TO-END DELAY AND PACKET LOSS RATE (PLR).

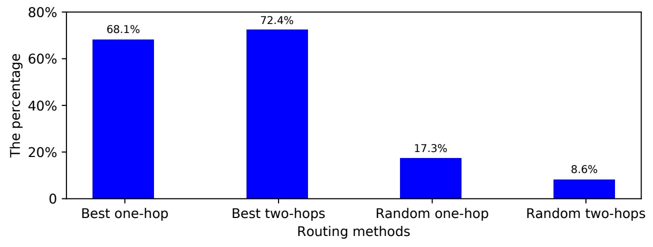
(Delay(ms), PLR)	Shanghai	Guangzhou	Moscow
Shanghai	$(0, 0.00\%)$	$(28.2, 0.04\%)$	$(262.1, 25.24\%)$
Guangzhou	$(27.6, 0.04\%)$	$(0, 0.00\%)$	$(245, 0.68\%)$
Moscow	$(261.5, 25.28\%)$	$(245, 0.75\%)$	$(0, 0.00\%)$

path. This high packet loss rate could result in extra 100ms delay due to retransmission mechanism. Surprisingly, if the packets first reach Guangzhou before arriving at Moscow, its delay would be less than the direct path delay between Shanghai and Moscow. The phenomenon that the indirect paths outperform direct paths in terms of end-to-end delay is the so-called triangle inequality violations (TIV), which has been widely reported in Internet [8], [9].

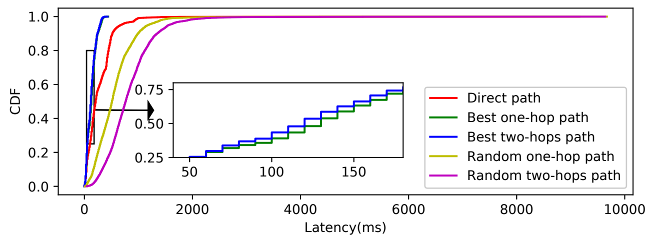
Based on the above discovery, some overlay routing methods have been proposed to steer traffic along the indirect but lower latency paths [10]–[12]. However, these methods have two shortcomings. First, these methods focus on constructing overlay routing over end hosts which are bundled with specific distributed applications, e.g., VoIP [13]. For those non-overlay applications or simple interdomain unicast session, the above methods are not applicable. Second, these overlay indirect routing methods are usually scaled poorly. Although the work in [11] has reduced the per-node communication from $O(n^2)$ to $O(n^{1.5})$ where n is the number of nodes in the overlay network, it still faces the scalability problem considering the scale of Internet.

In this paper, we study how to leverage the TIV to provide low-latency indirect routing paths for any pair of nodes across the WAN in a transparent way, which means that this service is provided by the network core and could be used by any applications without deploying overlay routing protocols. There are two main challenges in achieving this goal: *1) how to implement this routing strategy considering compatibility and feasibility.* The Internet has been evolving for decades, leaving so many legacy infrastructures including physical devices and software protocols. Thus, the new function deployment on the Internet must consider feasibility and compatibility, i.e., incremental deployment and no conflicts with other protocols. *2) how to pick up the correct indirect paths considering scalability and efficiency*. The dynamic network state may make the selected indirect paths fail in the future. Thus, we need to monitor the network and find the low-latency indirect paths continuously. However, the number of ASes could be as large as tens of thousands such that we have to achieve the goal in a distributed way.

To tackle the above two challenges, we propose the **SR**v6 **U**nderlay **F**ederation (SRUF) to provide low-latency routing service in the WAN. Segment routing over IPv6 (SRv6) is a new waypoint routing protocol which steers traffic through one or multiple waypoints along the route from a source to a destination [14]. The routing paths between any two neighbor waypoints are generated by existing routing protocols (refer to details in Sec. II-C). SRUF proposed in this work is composed of two main parts: a searching algorithm to find the low-latency paths in the control plane and an implementation method enabled by SRv6 in the forwarding plane. In the first part, some ASes will collaborate to form the SRUF. Through our highly scalable method, any pair of ASes in SRUF could find several candidate indirect low-delay paths with one or two relay nodes. In the second part, after finding the low-latency alternative path, the corresponding SRv6 instructions are inserted into the packet header when the SRUF node receives the packets. With the support of SRv6, the traffic could be routed along the selected path without modifying any legacy routing protocols.



(a) The percentages of indirect paths whose latencies are lower than them of direct paths under different methods.



(b) The CDFs of the latencies with different routing methods.

Fig. 2. Important observations: Comparisons among direct path, one-hop routing and two-hop routing.

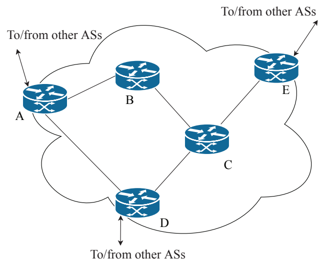
The contributions of this work are summarized as follows:

- **Compatibility:** SRUF could find low-latency routing paths in the core of wide area network without any modifications in deployed BGP and are transparent to applications (Sec. III).
- **Feasibility:** SRUF is open to accept new ASes to join in as long as the virtual network coordinates are updated and synchronized. An enhanced method further relaxes the restriction that all SRUF nodes should form a $\sqrt{n} \times \sqrt{n}$ grid in virtual network coordinates (Sec. IV).
- **Scalability:** Our method ensures that each node in SRUF only probes $O(\sqrt{n})$ nodes and disseminates its measurement results to other $O(\sqrt{n})$ nodes, empowering SRUF to support a large number of ASes (Sec. IV).
- **Efficiency:** We evaluate SRUF using numerical experiments and emulation experiments with real-world datasets. The experiment results demonstrate that SRUF can effectively reduce the average latency by $5.4\% \sim 58.9\%$ (Sec. VI).

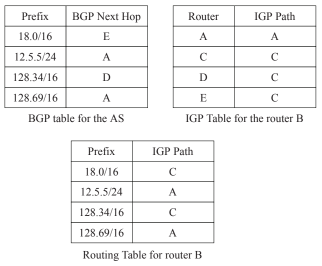
II. Motivation and Preliminaries

A. Observation

Though triangle inequality violations could be leveraged to reduce network latency, it still faces the challenge of selecting proper intermediate nodes. To address this issue, we analyze a public latency trace of Seattle network [15]. This trace contains 99 nodes and has latencies for every pair of nodes. We call the indirect path with one intermediate node as one-hop routing and two intermediate nodes as two-hops routing. We compare



(a) The overview of the illustrative example.



Prefix	BGP Next Hop
18.0/16	E
12.5.5/24	A
128.34/16	D
128.69/16	A

BGP table for the AS

Router	IGP Path
A	A
C	C
D	C
E	C

IGP Table for the router B

Prefix	IGP Path
18.0/16	C
12.5.5/24	A
128.34/16	C
128.69/16	A

Routing Table for router B

(b) BGP table, IGP table and the final routing table at Router B.

Fig. 3. Examples of interdomain and intradomain routing protocols. All routers run iBGP and an intradomain routing protocol. Border Routers A, D, and E also run eBGP to interconnect other autonomous systems [17].

two different methods, i.e., select the intermediate nodes randomly and select the best intermediate nodes. Fig. 2(a) and Fig. 2(b) show that both the best one-hop routing and the best two-hops routing posses lower latency than direct path in high probability. 68.1% best one-hop routing paths and 72.4% best two-hops routing paths have lower latencies. However, random one-hop routing and random two-hops routing have less than 20% probability to outperform the direct paths. Besides, Fig. 2(b) also shows that the indirect paths with two hops could perform better than one hop indirect path in some cases.

Above important observations show that intermediate nodes selection plays crucial role in reducing network latency. However, finding the best indirect paths is not easy because of lacking the whole view of network latencies. Considering that the Internet is composed of tens of thousands of ASes [16], probing all nodes in the network will lead to high overheads. Thus, we need to find a method to balance the optimality and scalability. On the other hand, we hope that the low latency routing service through indirect paths should be transparent to applications and provided by the network core. Our solution address all above concerns, which is introduced in detail in Sec III and IV.

B. Preliminary I: Internet Routing

The Internet is composed of billions of nodes, which is a huge distributed system spreading all over the world. To manage this huge system efficiently, it has been divided into many domains, i.e., autonomous systems (ASes). There are two mainly different kinds of routing protocols including the interior gateway protocol (IGP) and the exterior gateway protocol (EGP). IGP works in the inside of an AS and is used to exchange routing information between routers within the AS. EGP is a protocol for exchanging routing information among ASes. At now, BGP becomes the most important exterior gateway protocol. Fig. 3 gives an example of internet routing. Routers A, B, C, D and E belong to the same AS. Routers A, D and E run eBGP session with other ASes to exchange routing information and broadcast the learned routing information to other routers inside the AS. Each router in the AS will get a BGP table just as shown in the top left of Fig. 3(b). On the other hand, Router B will learn how to reach other routers inside the AS through IGP and get an IGP table (see the top right of Fig. 3(b)). Finally, Router B will combine BGP table and IGP table together and generate the final routing table which indicates the next-hop.

C. Preliminary II: Segment Routing over IPv6

The key idea of segment routing is to break a routing path into multiple segments, i.e, subpaths. There are two different data planes to implement segment routing including MPLS and IPv6. In this paper, we select SRv6 to implement our system as SRv6 can coexist with pure IPv6 network. Fig. 4 gives an illustrative example of SRv6. Host-1 and Host-2 address packets to Host-3. The packets from Host-1 and Host-2 will be first forwarded to the default gateway, i.e., Router 1. As introduced in Section II-B, the routing protocols will build routing paths on routers through configuring routing tables. Without loss of generality, we assume that three routing paths $\{R1 \rightarrow R2,\ R2 \rightarrow R3,\ R1 \rightarrow R3\}$ have been built by default routing protocols. R1 will forward all packets whose destination is Host-3 along the path $(R1 \rightarrow R3)$. However, if we want to balance the link utilization by steering the packets from Host-2 along $(R1 \rightarrow R2 \rightarrow R3)$, we could use SRv6 technology to add segment routing headers (SRH) into the packets. As shown in the top part of Fig. 4, for the packet from Host-2, R1 will add an SRH into it which is composed of two segment lists and an index variable, SL, indicating the current activated segment. Meanwhile, the destination address of the packet will be replaced by R2's address. When R2 received the packet, it will decrease the value of SL and use the activated segment list to replace the destination address, then the packet will go to R3 along $R2 \rightarrow R3$. Similarly, R3 will also decrease SL and forward the packet to the activated segment list, i.e., H3. Above example indicates that SRv6 could be used to steer traffic flexibly by combining the existed routing paths in demands.

In fact, such ability to appoint routing flexibly promises that SRv6 will bring new network programmability to SDN. Different from the clean-and-slate SDN protocol, OpenFlow [18], [19], which changes routing by rewriting routing tables in every switch devices along the path, SRv6 only needs to upgrade several crucial switch devices[1]. Literatures [14] and [21] showed that SRv6 has more advantages than OpenFlow in wide area network.

III. SRUF PHILOSOPHY

A. Label Technology and One-hop Routing

The success of Internet comes from its layer architecture. Each layer focuses on realizing its functions and provides its services to other layers through standard APIs, which are usually represented by labels in packet header. One classical application of label technology is multi-protocol label switching (MPLS), which is a routing technique that steers packets from one node to the next based on the labels rather than

[1]Any SRv6 enabled router could parse pure IPv6 packets. Besides, the network is evolving from IPv4 to IPv6 [20]; hence, in this paper, we assume that the network supports IPv6.

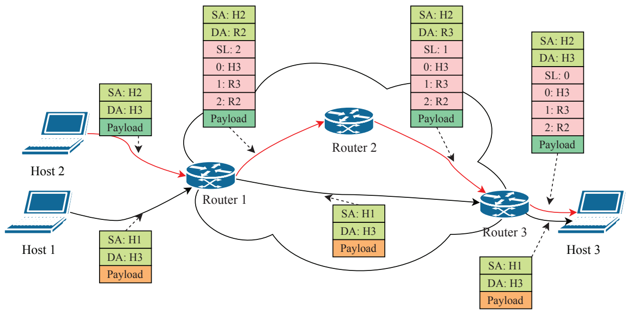


Fig. 4. An illustrative example of SRv6.

network addresses. This label technology brings many benefits to guarantee network performance.

One-hop source routing is first proposed in [22] to improve the reliability of Internet paths. This work showed that a path failure could be quickly recovered by routing indirectly through a small set of randomly chosen intermediaries. It inserts an extra label in the packet header to steer the packet to the selected intermediary node for avoiding traversing the failed links. Note that the extra label, i.e., the IP address of the intermediary node, will call the basic routing service provided by the bottom IP layer automatically.

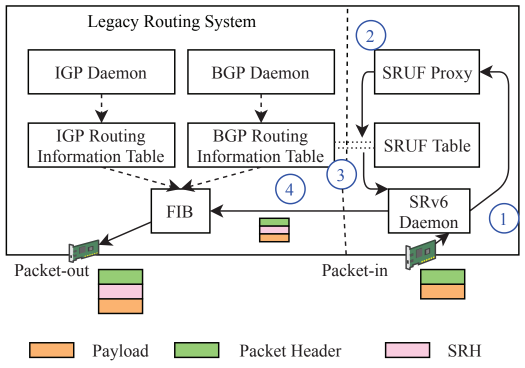
The existence of triangle inequality violations (TIV) and the idea of one-hop source routing motivates us to design a compatible method for the network core which provides low-latency service in inter-domain routing based on current BGP infrastructure.

B. Overview of SRv6 Underlay Federation

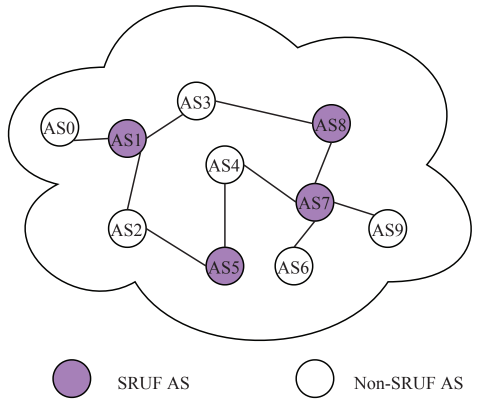
Our goal in this paper is to address the challenges of providing low-latency routing services in Internet without modifying legacy inter-domain routing infrastructures. Our solution, called SRUF, is an SRv6-based underlay federation routing system that substitutes the default direct routing paths with the selected indirect low-latency delay paths. Fig. 5 gives an overview of the SRUF including its architecture in single AS (see Fig. 5(a)) and its incremental deployment in Internet (see Fig. 5(b)). All the members in SRUF will work together to find the low-latency indirect path.

As shown in Sec. II-B and Fig. 5(a), in the legacy routing system, routers run IGP and BGP simultaneously. These protocols will update their routing information tables in real-time in order to catch up the network dynamics. Finally, these routing tables will be combined together to generate forward information base (FIB). When a router receives a new packet, it will forward the packet to correct port through looking up the FIB. Here, we assume that the BGP protocol is configured to select the shortest AS path. Thus, in AS0, any packet forwarded to AS9 will be routed along ($AS0 \rightarrow AS1 \rightarrow AS3 \rightarrow AS8 \rightarrow AS7 \rightarrow AS9$).

SRUF never interacts with the control plane of legacy routing systems, it just parses the BGP routing information table and leverages SRv6 to change the routing destination. There are three main components in SRUF, including the SRv6 daemon, the SRUF proxy and the SRUF table.



(a) The deployment of SRUF in single AS.



(b) The deployment of SRUF in Internet.

Fig. 5. The overview of SRUF

- The SRv6 daemon: this component is responsible for getting the destination address by parsing the packet header and assigning new routing path by encapsulating new SRv6 segment label. There is already an open-source software of SRv6, thus in this paper we do not focus on the implementation of SRv6 daemon [23], [24].
- The SRUF table: this component is the crucial part of SRUF. Table II gives an example, it records the alternative low-latency paths. The first attribute reflects the destination AS. The second attribute records the IPv6 addresses of the ASes on the alternative path which is used to insert the segment routing header to steer the packet along the low-latency path. Besides, to fresh the network state in time, we set TTL values for each alternative path. The unit of TTL could be set with per one minute, per 10 minutes or other proper intervals.
- The SRUF proxy: this component first parse the packet passed by SRv6 daemon to get the next-hop address and then look up the SRUF table to get the proper alternative path. Finally, it will tell the SRv6 daemon how to encapsulate the packet. Besides, the SRUF proxy is responsible for updating SRUF table.

SRUF is a distributed system, where any AS can participate. In SRUF, each AS can ping to any other AS to measure the delay of the default direct path which is decided by

TABLE II
AN ILLUSTRATIVE EXAMPLE OF SRUF TABLE IN $AS1$.

Destination	Alternative Path	TTL
$AS5$	$\emptyset$	1
$AS7$	$IP(AS5)$	3
$AS8$	$IP(AS5) \rightarrow IP(AS7)$	2

the legacy routing system. These delay measurement results will be disseminated into the selected SRUF members (the selection strategy is explained in Sec. IV). Leveraging these measurement results, we can easily find the low-latency indirect path between a pair of ASes and record them in a table, i.e., SRUF Table. However, there are several challenges in constructing SRUF Tables at each SRUF member considering the overhead and scalability. These challenges are introduced in Sec. III-C and solved in Sec. IV.

Here, we give an example to illustrate the workflow of SRUF clearly. As shown in Fig. 5, AS1, AS5, AS7 and AS8 are the members of SRUF. These four ASes have found the low-latency indirect paths among each other. Without loss of generality, we assume that AS1 finds that its delay measurement results satisfy $t_{AS1 \rightarrow AS5} + t_{AS5 \rightarrow AS7} < t_{AS1 \rightarrow AS7}$. Next, we present the journey of a packet originating from AS0 to AS9 in Fig. 5 as follows,

1) When the packet reaches to AS1 which is a member of SRUF, it will first be intercepted by the SRv6 daemon at $AS1$. If the packet is an SRv6 packet, SRv6 daemon will decrease the value of SL, activate corresponding segment and then pass it to the SRUF proxy. If the packet is a pure IPv6 packet, it will be moved to the SRUF proxy directly.
2) The SRUF proxy will look up the BGP routing information table and find that the AS path for this packet is $(AS3 \rightarrow AS8 \rightarrow AS7 \rightarrow AS9)$. Additionally, it finds that there are two SRUF members, $\{AS7, AS8\}$, and the farthest one is $AS7$.
3) Through looking up the SRUF table, it finds that the delay between $AS1$ and $AS7$ could be reduced by an indirect path $AS1 \rightarrow AS5 \rightarrow AS7$. Thus, it adds the address of AS5, i.e., $IP(AS5)$, into the packet's SRH.
4) The new encapsulated packet will be forwarded to $AS5$ according to the FIB.

From above example, we can see that SRUF uses routers in the network core to intercept the packets and leverage SRv6 to change the routing path without modifying any FIBs generated by legacy routing protocols; hence, SRUF naturally possesses compatibility to the legacy Internet infrastructure. The crucial point of SRUF is that how to find the correct low-latency indirect paths. Besides, the feasibility and scalability of SRUF relies on the construction of SRUF Table. The challenges in constructing the SRUF table will be shown in Sec. III-C.

C. The Challenges of Constructing SRUF Table

Note that each SRUF member cannot find the low-latency indirect paths between it to all other members with only knowing its measurement results. For example, if the low-latency indirect path between A and B is $(A \rightarrow C \rightarrow B)$, neither A or B could find this path since A does not know t_{CB} and B does not know t_{AC}. To address the above problem, we can make every SRUF member node not only periodically monitor its links to all other $n-1$ member nodes in SRUF, but also disseminate its measurement results with $n-1$ entries to them. Such method generates $O(n^2)$ per-node probing and disseminating overhead. This method ensures that each node knows the delay between any pair of nodes; hence being able to finding the best low-latency indirect path for any pair of nodes. The probing overhead and disseminate overhead grow linearly with the number of probed nodes. Thus, having every node continuously monitor all of the other nodes is neither feasible nor desirable for large-scale networks.

One alternative method is to make every node probe a set of nodes and just disseminate its probing results to several selected nodes. Then the selected member nodes could find low-latency paths for the pair of nodes which have sent them probing results. For example, node A and node B will probe a set of nodes independently. Assume that both A and B have probed node C. We can make A and B send their probing results to C, then C could calculate the delay of the indirect path $A \rightarrow C \rightarrow B$. However, this simple idea will be hardly feasible until the following challenges have been solved.

- The first challenge is that every member node is unaware of which member node it should probe. One method is to make every node probe all other nodes at the cost of high bandwidth overhead. The other method is to make every node probe a set of nodes randomly [22]. However, this method cannot ensure that the intersection of two probing sets is nonempty for any node pair and usually cannot find candidate indirect paths for that node pair. In summary, we need a deterministic approach that could find as more candidate paths for every node pair as possible.
- The second challenge comes from selecting a set of member nodes to collect the probing results. The constraint is that we need to ensure that any pair of member nodes have at least one common node receiving their probing results. Though we could make a central node to receive all probing results, it will suffer the performance bottleneck and the single point of failure. We need to find a distributed approach that could find common rendezvous nodes for each node pair.
- The third challenge is to enlarge the set of candidate paths to select the best low-latency path. One method to realize this goal is to increase the probing set of each node while it will increase the overhead. The analysis in Sec II-A shows that two relay nodes could significantly increase the possibility to get low-latency paths.

IV. LOW-LATENCY PATH SELECTION

A. Virtual Network Coordinate

From Sec. III-C, we need to find a deterministic approach to decide the set of probing nodes for each member of SRUF. To track and probe these member nodes efficiently, we build

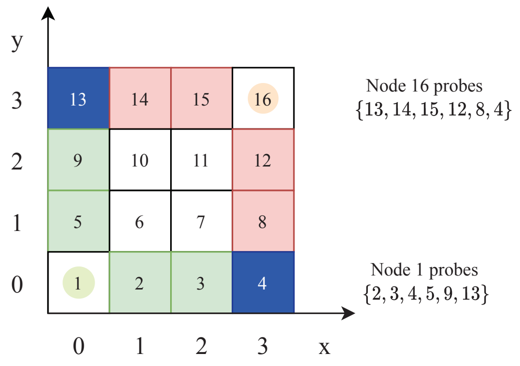


Fig. 6. The virtual network coordinate in SRUF.

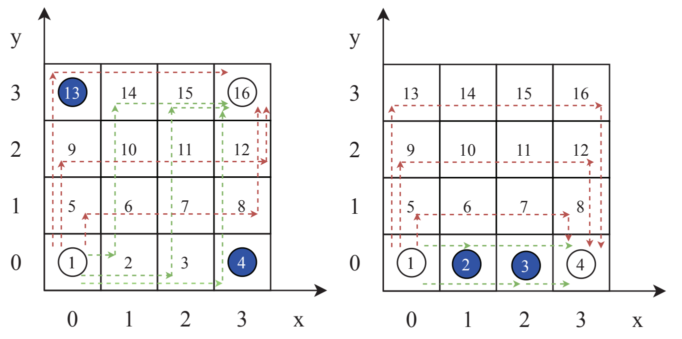
a virtual network coordinate system. Assume that we have $\sqrt{n} \times \sqrt{n}$ SRUF members $\{q_1, q_2, ..., q_n\}$ where $\sqrt{n}$ is an integer (the more general case will be introduced in Sec. V). We place these SRUF members into a $\sqrt{n} \times \sqrt{n}$ grid. Without loss of generality, we assume that the node q_i fills the i^{th} grid cell in a managed way. It needs to be emphasized that the detail manage method is out of scope of this paper. Actually, we can manage these nodes through network configuration or using P2P registration.

B. Candidate Paths

*1) **Probe and Disseminate**:* After embedding all the SRUF members into the virtual network coordinate, we find that this coordinate has some structure features. For any node q_i whose position is denoted by (x_i, y_i), let $S(q_i)$ denote the set that contains $2\sqrt{n} - 2$ nodes in row x_i and column y_j except itself. For another node q_j whose position is (x_j, y_j), we can easily know that $S(q_i)$ and $S(q_j)$ share two common nodes in positions (x_i, y_j) and (x_j, y_i) if q_i and q_j are in different rows and columns; otherwise, q_i and q_j share $\sqrt{n} - 2$ nodes in the same row or column except themselves. For example, in Fig. 6, q_1 and q_{16} are in different rows and columns. $S(q_1) = \{q_2, q_3, q_4, q_5, q_9, q_{13}\}$, $S(q_{16}) = \{q_4, q_8, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}\}$, so they share two common nodes $\{q_4, q_{13}\}$. q_1 and q_{13} are in the same column and $S(q_{13}) = \{q_1, q_5, q_9, q_{14}, q_{15}, q_{16}\}$. We can see that $S(q_1)$ and $S(q_{13})$ share $\sqrt{16} - 2$ nodes, i.e., $\{q_5, q_9\}$. With above definition about $S(q_i)$, we propose the probe strategy as follows,

Probe Strategy: For any node q_i, let it probe all the nodes in $S(q_i)$. Since $S(q_i)$ and $S(q_j)$ share 2 or $\sqrt{n}-2$ nodes, this probe strategy ensures that the intersection of two probing sets for every node pair will never be empty. Thus, the first challenge in Sec. III-C is solved. Additionally, this probe strategy ensures that every node is probed balancedly, i.e, every node q_i is probed by $2\sqrt{n} - 2$ nodes in $S(q_i)$.

In probe phase, node q_i will acquire a link state table recording the latency measurement results between it and all nodes in $S(q_i)$. To find the possible low-latency indirect path between q_i and q_j, there should be at least one node receive the link state tables from both q_i and q_j. We call such nodes that are responsible for finding the alternative paths as **rendezvous nodes**. As shown in the second challenge



(a) Case 1: the candidate paths between node 1 and node 16. (b) Case 2: the candidate paths between node 1 and node 4.

Fig. 7. The examples of candidate paths between a pair of nodes in two different cases.

in Sec. III-C, we could select a central rendezvous node to receive link state tables from all nodes. This approach will consume $O(\sqrt{n})$ bandwidth per-node since each node probes $O(\sqrt{n})$ nodes. However, this approach relies heavily on the central rendezvous node which may cause the single-point failure. Besides, finding the candidate paths for all node pairs in a central rendezvous node would incur the performance bottleneck. Another approach is to make every node broadcast its link state table to all other $n - 1$ nodes which will consume $O(n^{1.5})$ bandwidth per-node [11]. This approach will also bring too much redundant information since every node can calculate the candidate paths for any pair of nodes. Combing above two approaches together, we propose our own disseminate strategy as follows,

Disseminate Strategy: For any node q_i, let it disseminate its latency measurement results to all the nodes in $S(q_i)$. In this way, every node q_i acts as a rendezvous node for the other $2\sqrt{n}-2$ nodes. Thanks to the probe strategy, any pair of nodes have 2 or $\sqrt{n}-2$ rendezvous nodes. Thus, the second challenge in Sec. III-C is solved. Since there are $O(\sqrt{n})$ nodes in $S(q_i)$, node q_i will consume $O(\sqrt{n} \times \sqrt{n}) = O(n)$ bandwidth.

*2) **Path Selection**:* After probing and disseminating, each rendezvous node is prepared to calculate the candidate paths. For a pair of node q_i and q_j, there are two cases for their relationship. We introduce the path selection in these two cases respectively.

Case 1: Node q_i and node q_j are in different rows and columns where $x_i \neq x_j$ and $y_i \neq y_j$. There are $2\sqrt{n} - 2$ candidate paths from q_i and q_j, including $\sqrt{n} - 1$ paths (q_i, q_a, q'_a, q_j) and $\sqrt{n}-1$ paths (q_i, q_b, q'_b, q_j), where the relay nodes q_a, q'_a, q_b and q'_b are in positions (x_a, y_i), (x_a, y_j), (x_i, y_b) and (x_j, y_b). Here, $x_a \in \{1, 2, ..., \sqrt{n}\} - \{x_i\}$, and $y_b \in \{1, 2, ..., \sqrt{n}\} - \{y_i\}$.

There are two common rendezvous nodes in position (x_j, y_i) and (x_i, y_j) which are responsible for calculating the best low-latency path. For the rendezvous nodes, they will receive the latency measurement results from all nodes in $S(q_i)$ and $S(q_j)$; hence, they know the latency of each hop in the candidate paths. After calculating the latencies of all candidate paths, the rendezvous node will select the best one and disseminate this path to q_i and q_j. For example, in

Algorithm 1 Finding backup paths at any node q_k in SRUF

```
Input : Node $q_k$ and the virtual network coordinate of all SRUF
    nodes.
Output : A set of low-latency paths for some node pairs.
 1: for Any pair of nodes $(q_i, q_j)$ in $S(q_k)$ do
 2:     Let $P$ denote the candidate indirect paths between $q_i$ and $q_j$
        and initialize it as $\emptyset$
 3:     if $q_i$ and $q_j$ are not in the same row and column then
 4:         for $x_a$ in $\{1, 2, ..., \sqrt{n}\} - \{x_i\}$ do
 5:             Add path $(q_i, q_a, q'_a, q_j)$ into $P$ where $q_a = (x_a, y_i)$ and
                $q'_a = (x_a, y_j)$
 6:         for $y_b$ in $\{1, 2, ..., \sqrt{n}\} - \{y_i\}$ do
 7:             Add path $(q_i, q_b, q'_b, q_j)$ into $P$ where $q_b = (x_i, y_b)$ and
                $q'_a = (x_j, y_b)$
 8:         Select the path with the lowest latency from $P$ and notify
            this path to $q_i$ and $q_j$
 9: for Any node $q_i \in S(q_k)$ do
10:     Let $P$ denote the candidate paths between $q_i$ and $q_k$
11:     if Nodes $q_i$ and $q_k$ are in the same row then
12:         for $x_a$ in $\{1, 2, ..., \sqrt{n}\} - \{x_i, x_k\}$ do
13:             Add path $(q_k, q_a, q_i)$ into $P$ where $y_a = y_k$
14:         for $y_b$ in $\{1, 2, ..., \sqrt{n}\} - \{y_k\}$ do
15:             Add path $(q_k, q_b, q'_b, q_i)$ into $P$ where $x_b = x_k$ and $x'_b =$
                $x_i$
16:     if Nodes $q_i$ and $q_k$ are in the same column then
17:         for $y_a$ in $\{1, 2, ..., \sqrt{n}\} - \{y_i, y_k\}$ do
18:             Add path $(q_k, q_a, q_i)$ into $P$ where $x_a = x_k$
19:         for $x_b$ in $\{1, 2, ..., \sqrt{n}\} - \{x_k\}$ do
20:             Add path $(q_k, q_b, q'_b, q_i)$ into $P$ where $y_b = y_k$ and $y'_b =$
                $x_i$
21: Save the lowest latency path from $P$ in $q_k$ and notify the path
    to $q_i$
```

Fig. 7(a), node q_1 and node q_{16} are in different rows and columns. Node q_4 and node q_{13} are the rendezvous nodes that are responsible for calculating the latencies of the candidate paths between q_1 and q_{16}. Here, we take one candidate path $(q_1, q_2, q_{14}, q_{16})$ as an example. The one-hop latencies $t_{q_1q_2}$ and $t_{q_2q_{14}}$ will be disseminated to q_4 by q_2, while the one-hop latency $t_{q_{14}q_{16}}$ will be disseminated to q_4 by q_{16}; hence, q_4 could calculate the latency of $\{t_{q_1q_2}, t_{q_{14}q_{16}}\}$. Note that q_{13} also can calculate the latency of $\{t_{q_1q_2}, t_{q_{14}q_{16}}\}$ since it can know $t_{q_1q_2}$ from q_1 and $\{t_{q_1q_2}, t_{q_{14}q_{16}}\}$ from q_{14}. Consequently, the two rendezvous nodes can calculate the latencies of all $2\sqrt{n} - 2$ candidate paths and find the best path among them. Finally, the best path will be notified to q_1 and q_{16}.

Case 2: In case two, q_i and q_j are in the same row or column where $\{x_i \neq x_j, y_i = y_j\}$ or $\{x_i = x_j, y_i \neq y_j\}$. There are $2\sqrt{n}-3$ candidate paths from q_i and q_j, including $\sqrt{n}-2$ paths (q_i, q_a, q_j) and $\sqrt{n} - 1$ paths (q_i, q_b, q'_b, q_j). For the former $\sqrt{n} - 2$ paths, the relay node q_a can be any node in the same row or column with q_i and q_j but not themselves. For the other $\sqrt{n} - 1$ paths, the relay nodes q_b and q'_b are in the positions (x_i, y_b) and (x_j, y'_b) where $y_b = y'_b \in \{1, 2, ..., \sqrt{n}\} - \{y_i\}$ if q_i and q_j are in the same row. If q_i and q_j are in the same column, the relay nodes q_b and q'_b are in the positions (x_b, y_i) and (x'_b, y_j) where $x_b = x'_b \in \{1, 2, ..., \sqrt{n}\} - \{x_i\}$. Different from case 1, q_i and q_j can direct calculate the latency of these $2\sqrt{n}-3$ candidate paths locally. See the example in Fig. 7(b), for path (q_1, q_2, q_4), q_1 would get $t_{q_1q_2}$ by its own probing result and get $t_{q_2q_4}$ from q_2. For path (q_1, q_5, q_8, q_4), q_1 could get $\{t_{q_1q_5}, t_{q_5q_8}\}$ from q_5 and $t_{q_8q_4}$ from q_4. Finally, both q_1 and q_4 can find the best candidate path locally.

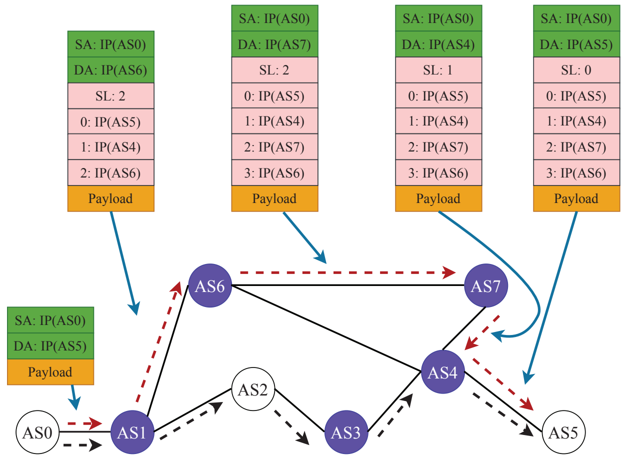


Fig. 8. An illustrative example of inserting SRHs in SRUF.

From above two cases, we can conclude that when rendezvous node q_k calculates the best candidate paths between $q_i \in S(q_k)$ and $q_j \in S(q_k)$, it will only need to concern the case that q_i and q_k are not in the same row or column. If $q_i, q_j \in S(q_k)$ and they are in the same row or column, the candidate paths can be calculated by themselves. In Fig. 7(b), node 1 and node 4 are in the same column, rendezvous node 2 does not need to calculate the candidate paths for them. Besides, node q_k is also responsible to calculate the candidate paths from it to other nodes in $S(q_k)$. Thus, we divide the operations in rendezvous node q_k into two rounds and summarize them with details in Algorithm 1.

From above descriptions, we can conclude that given any pair of nodes, our method could generate $2\sqrt{n} - 3$ candidate indirect paths if the two nodes are in the same row or column; otherwise, $2\sqrt{n}-2$ candidate indirect paths will be generated.

C. SRv6 Header Insertion

Each node q_k in SRUF will finally record the alternative paths toward all other nodes in its SRUF table. When a packet is intercepted by SRUF proxy in q_k, it will first look up the BGP table to get the default BGP path. Then it will find the last SRUF node in the BGP path and insert its corresponding IPv6 address into the SRH. To introduce the SRv6 header insertion clearly, we give an example in detail. As shown in Fig. 8, a packet from AS0 hopes to reach AS5 and the default BGP path is $(AS0, AS1, AS2, AS3, AS4, AS5)$. We assume that $\{AS1, AS3, AS4, AS6, AS7\}$ has joined SRUF and have built the SRUF tables. When the packet arrives at AS1, the SRUF proxy will intercept it and find that the last SRUF node in the default BGP path is $AS4$. Then AS1's SRUF proxy will lookup the low-latency path between AS1 and AS4, which is $(AS1, AS6, AS4)$. Thus, it will add the IPv6 addresses of $\{AS5, AS4, AS6\}$ into the packet header. When AS6 receives the packet, its SRv6 daemon will first parse the packet and decrease the value of SL and find that the next-hop destination

Algorithm 2 Inserting Segments into Packet Header in SRUF-enabled AS_i

```
Input : The packet $P$ intercepted by SRv6 daemon
Output : The new encapsulated packet $P'$.
1:  SRv6 daemon parse $P$, get the value of segment index $SL$
2:  if $P$ does not have SRH then
3:      Set the value of segment index with 0, i.e., $SL = 0$
4:  Look up the BGP table to get the default AS path and get the farthest SRUF-enabled AS, i.e., $AS_j$
5:  Look up the SRUF table to get the alternative path $p_{ij}$ between $AS_i$ and $AS_j$
6:  if $p_{ij}$ has one relay node $AS_k$ then
7:      Insert segments $SL : IP(AS_j)$, $SL + 1 : IP(AS_k)$ into header of $P$ and let $SL = SL + 1$
8:  else if $p_{ij}$ has two relay nodes $AS_k$, $AS_h$ then
9:      Insert segments $SL : IP(AS_j)$, $SL+1 : IP(AS_k)$, $SL+2 : IP(AS_h)$ into header of $P$ and let $SL = SL + 2$
10: else if $p_{ij}$ is $\emptyset$ then
11:     Pass
```

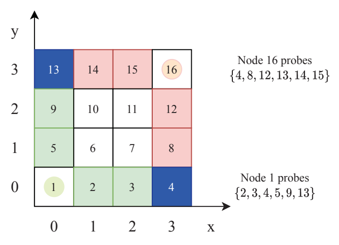
is AS4. SRv6 daemon will pass this destination information to the SRUF proxy. The SRUF proxy find that there will be a low-latency indirect path between AS6 and AS4, i.e., (AS6, AS7, AS4), then it will insert AS7's IPv6 address into the SRH and update the value of SL. Similarly, AS7 and AS4 will do the same operations. Finally, the packet will reach its destination. We summarize the procedures of inserting SRH in Algorithm 2.

V. Enhanced Method

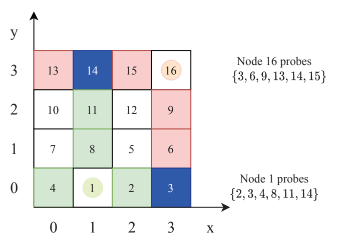
Using the aforementioned method to find the alternative low-latency paths still face two problems. 1) The number of SRUF nodes may not be just $\sqrt{n} \times \sqrt{n}$, thus cannot form a perfect square like Fig 7. 2) Each node always probes the same set of nodes, which may miss some better paths to other nodes. The above two problems could be solved by our adaptive virtual network coordinate.

A. SRUF with Any Number of Nodes

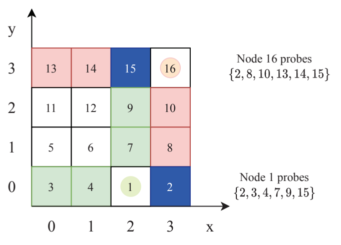
If the SRUF has n nodes and $\sqrt{n}$ is not an integer, we could still form a $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ grid and place those n nodes into the grid. There will be $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil - n$ empty cells whose positions are $\{(n\%\lceil \sqrt{n} \rceil, \lceil \sqrt{n} \rceil), (n\%\lceil \sqrt{n} \rceil + 1, \lceil \sqrt{n} \rceil), ..., (\lceil \sqrt{n} \rceil - 1, \lceil \sqrt{n} \rceil)\}$. To fill the grid, we copy the nodes in positions $\{(n\%\lceil \sqrt{n} \rceil, \lceil \sqrt{n} \rceil - 1), (n\%\lceil \sqrt{n} \rceil + 1, \lceil \sqrt{n} \rceil - 1), ..., (\lceil \sqrt{n} \rceil - 1, \lceil \sqrt{n} \rceil - 1)\}$ to the empty nodes. Then the virtual network coordinates will be filled and the proposed method in Sec. IV could be leveraged. Here we give an example to illustrate this enhanced method with more details. Assume that there are thirteen SRUF nodes $\{q_1, q_2, ..., q_{13}\}$ in Fig. 7(b)'s example, then positions $\{(1, 3), (2, 3), (3, 3)\}$ will be empty. To fill the 4×4 grid, we could copy q_{10} in $(1, 3)$, q_{11} in $(2, 3)$ and q_{12} in $(3, 3)$. Then the path $(1 \rightarrow 13 \rightarrow 16 \rightarrow 4)$ really means $(q_1 \rightarrow q_{13} \rightarrow q_{12} \rightarrow q_4)$. It should be noted that this completion method may generate more one-hop indirect paths. For example, the two-hops indirect path $(1 \rightarrow 13 \rightarrow 16 \rightarrow 12$ really means the one-hop indirect path $(q_1 \rightarrow q_{13} \rightarrow q_{12})$.



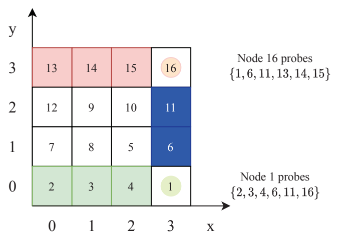
(a) The positions of all SRUF members in round 1.



(b) The positions of all SRUF members in round 2.



(c) The positions of all SRUF members in round 3.



(d) The positions of all SRUF members in round 4.

Fig. 9. Rotational sampling: change the positions of all the SRUF members iteratively in the virtual network coordinate in different time round.

B. Rotational Sampling

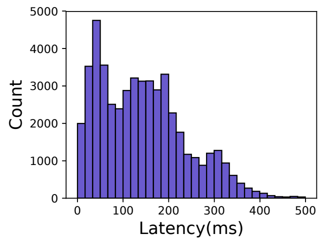
Though each node in SRUF only needs to probe $O(\sqrt{n})$ nodes which is much lower than probing all nodes, it may miss some better low-latency paths. To explore more candidate paths without increasing the overhead of each node, we propose rotational sampling in each round. In detail, rotational sampling changes the positions of all SRUF nodes in a regular way, that is, all nodes in the i^{th} row move $i + 1$ cells right in each new round. Fig. 9 gives an illustrative example of rotational sampling. In round 2, we move all nodes in the 0^{th} row one cell to the right, all nodes in the 1^{th} row two cells to the right, all nodes in the 2^{th} row three cells to the right and all nodes in the 3^{th} row four cells to the right. Then we will get $S(q_1) = \{q_2, q_3, q_4, q_8, q_{11}, q_{14}\}$ and $S(q_{16}) = \{q_3, q_6, q_9, q_{13}, q_{14}, q_{15}\}$. These two new sets will explore new indirect paths. This rotational sampling method in fact enlarge the set of candidate indirect paths and improves the possibility to find the better low-latency alternative path.

VI. Performance Evaluation

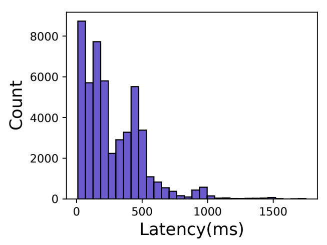
In this section, we first introduce the experiment setting and then evaluate the performance of SRUF in numerical experiments and emulation experiments.

A. Experiment Setting

1) AS-topology: We use two different AS-topologies to evaluate SRUF. The method to collect AS-level topology has been introduced in literature [16] and one public dataset could be downloaded in [25]. We select a part of this topology which has 6313 nodes as our large-scale topology in numerical experiment. As for the emulation experiment, we select a small-scale topology which has 42 nodes as our ground network from Internet Topology Zoo [26].



(a) Trace 1: The distribution of synthesized latency based on PlanetLab dataset.



(b) Trace 2: The distribution of synthesized latency based on Seattle dataset.

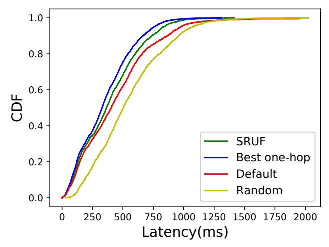
Fig. 10. The distributions of synthesized latency traces over all links of the large-scale AS-level topology in one time slot.

2) Link latency: There are no latency datasets in [25] and [26]; hence, we synthesis link latency dataset by ourselves based on the latency traces of Seattle network and PlanetLab network [15]. Both of the two traces have different time slices. Based on the theory of self-similarity in Internet packet delay [27], we assign the value sampled from the latency datasets to our AS-topologies randomly. Fig. 10 shows the distributions of the two synthesized traces in one time slice. It should be emphasized that the SRUF method could work regardless of the latency distribution.

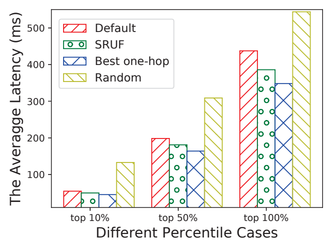
3) Compared methods: To evaluate the performances of our SRUF in finding the low-latency paths, we compare it with two detour routing methods. One is the random one-hop algorithm in literature [22], which aims for improving the robustness of the network. The other one is the best one-hop algorithm in literature [11]. Note that above two benchmark algorithms only have impacts on picking up alternative paths, they still need to collaborate with other modules in SRUF to provide low latency routing service in network core. In other words, the two compared methods only have differences in selecting the relay node. Also, we compare SRUF to the default BGP with preferring the minimum AS hops.

B. Numerical Result

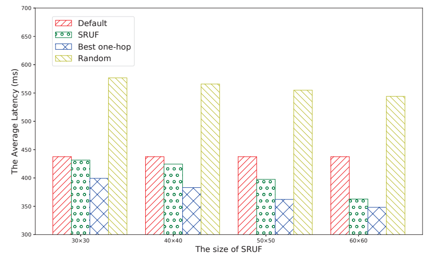
1) The performances of SRUF in static environment: Fig. 11 and Fig. 12 depict the experiment results in a static environment, where the latency of each link is unchanged. From these results, we can conclude that both SRUF and the best one-hop method can certainly reduce the latencies. We first set the size of SRUF is 60×60. Fig. 11(a) and Fig. 12(a) illustrate that best one-hop method could find the better alternative paths than SRUF. This is because that best one-hop method alway find the optimal alternative path with the whole view of network. The gap between the best one-hop method and our SRUF method in Fig. 11(a) is larger than that in Fig. 12(a). The reason behind this phenomenon is that the distribution of link latencies will influence the performance of SRUF. This phenomenon also indicates that SRUF is sure to work no matter what the link latency distribution is. The skew distribution like Fig. 10(b) will increase the probability of finding the better alternative paths in SRUF. Besides, we compare the average latency in different percentiles. As shown in Fig. 11(b) and Fig. 12(b), the SRUF method has little



(a) The comparisons of latency distributions with 60×60 SRUF size.

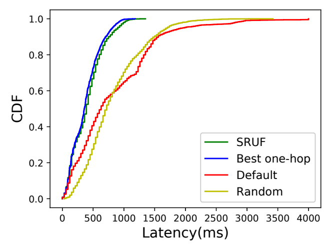


(b) The comparisons of latencies with 60×60 SRUF size.

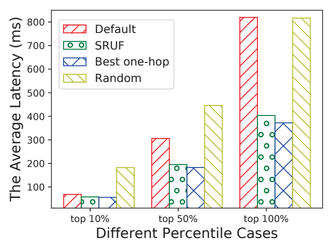


(c) The average latency comparisons with different SRUF sizes.

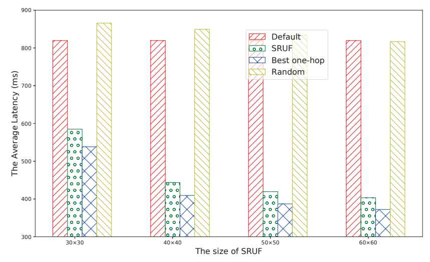
Fig. 11. The performances of different methods with synthesized trace 1.



(a) The comparisons of latency distributions with 60×60 SRUF size.



(b) The comparisons of latencies with 60×60 SRUF size.

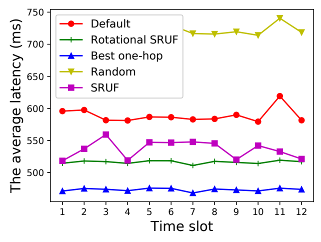


(c) The average latency comparisons with different SRUF sizes.

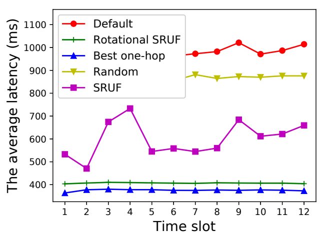
Fig. 12. The performances of different methods with synthesized trace 2.

improvement in top 10% link latency, but it can reduce the default average latency of top 50% by 8.7% and 36.2%, and reduce the default average latency of all links by 11.7% and 50.8%. This phenomenon indicates that it is easier for SRUF to optimize the paths in long tail latency part.

We furtherly evaluate the performances of SRUF with different sizes, i.e., $\{30 \times 30, 40 \times 40, 50 \times 50, 60 \times 60\}$. As shown in Fig. 11(c) and Fig. 12(c), compared to SRUF with 30×30 size, SRUF with 60×60 size can reduce the default average latency by 15.9% and 31%, respectively. This phenomenon satisfies the design of SRUF. In Sec. IV-B, we have proved that the SRUF can find $2\sqrt{n} - 2$ or $2\sqrt{n} - 3$ candidate paths in different cases. The larger size of SRUF



(a) The performances of different methods with trace-1.



(b) The performances of different methods with trace-2.

Fig. 13. The comparisons of the average latencies in dynamic environment.

can find more candidate paths and then find better low-latency paths.

2) The performances of SRUF in dynamic environment: In this experiment, we set 12 time slots in total and change the link latencies in each slot. We select 3600 nodes randomly to form a 60×60 SRUF. Additionally, we evaluate the performance of enhanced SRUF which is introduced in Sec. V-B. As shown in Fig. 13(a) and Fig. 13(b), our SRUF method can reduce the latencies by 7.0% and 40.72% comparing to the Default method respectively, however, these paths have much fluctuation as the default routing paths. This phenomenon may be caused by that always probing the same set of nodes lose the opportunity to find better paths. As a contrast, the enhanced SRUF with rotational sampling performs more stable and it can reduce the latencies by 12.4% and 58.9% against the Default method, respectively. In fact, the reason why rotation SRUF performs better is that it enlarges the set of candidate paths through probing dynamically.

C. Emulation Result

We conduct a small-scale emulation experiment to further evaluate our method. We use IPMininet [24] to implement a prototype system. This experiment emulates a small-scale real network topology as shown in Fig. 14 which has 42 nodes. Similar to the numerical experiment, we also synthesized link latencies with the Seattle trace. As shown in Fig. 15(a), our SRUF method and the best one-hop method have different bandwidth consumptions per-node. The gap between the two curves increases as the size of SRUF becomes larger. This phenomenon is consistent with our conclusion in Sec. IV-B that the per-node bandwidth overhead is $O(n)$ in SRUF and $O(n\sqrt{n})$ in best one-hop method. The 95^{th} percentile latency of the default routing is 1520 ms and our SRUF can reduce it by 16.4% reaching 1270 ms.

In Fig. 15(c), we evaluate the SRUF with different sizes. When the SRUF only have 2×2 members, its average delay is higher than the default method's. Meanwhile, we can see that the average latency of SRUF decreases as the SRUF size grows up. When the size of SRUF reaches 4×4, its average latency becomes lower than the default method's. In 6×6 SRUF, the average latency is 25.2% lower than it of the default method.

VII. Related Work

Existing solutions for providing low-latency path routing in the wide area networks include two categories: protocol-inside improvement and protocol-outside control.

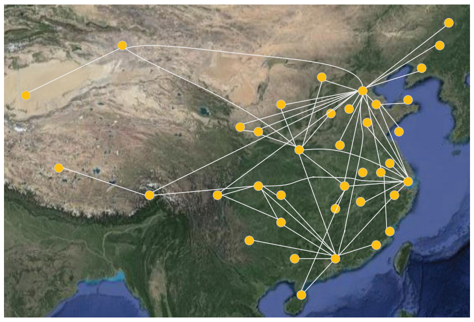
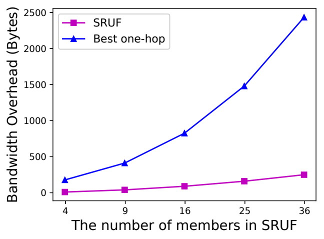
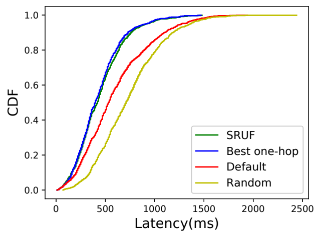


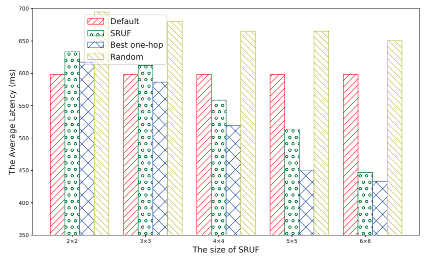
Fig. 14. The topology of China Telecom [26].



(a) The bandwidth consumption in one SRUF member node.



(b) The comparisons of latencies with 60×60 SRUF size.



(c) The comparisons of average latencies with different SRUF sizes.

Fig. 15. The performances of different methods in emulation experiments.

Protocol-inside improvement: As the infrastructure of inter-domain routing protocol, BGP is architecturally rigid. BGP requires direct neighbors to use the same protocol, which makes it more difficult for the Internet's routing infrastructure to change or evolve [28]. In fact, in 1999, literature [29] have showed that there were large number of alternative paths that had lower path latencies than the paths chosen by BGP. However, until 2020, performance-based BGP routing mechanisms are still in progress [30], [31]. The huge inertia of legacy inter-domain infrastructure makes any modification or improvement of BGP hard to move forward.

Protocol-outside control: To sidestep the performance-obliviousness of BGP routing, content providers build sophisticated control systems that use performance measurements to serve clients via low-latency options [32]. For example, Microsoft measured performance from clients to different servers and then use their DNS servers to redirect the client to the best server [33]. Google even use Espresso to route cloud traffic via its private WAN instead of public Internet to avoid traversing multiple ASes [34]. Above content providers' solutions mainly serve for their own traffic in private WANs, while our solution aims to provide general low-latency service for any kind of traffic through cooperation of multiple ASes in Internet. On the other hand, some distributed applications, e.g., VoIP and P2P system, use overlay routing in application layer to find low-delay paths. In [10], the authors implemented a latency-reducing routing overlay system named PeerWise. Literature [11] and [12] devoted to finding the best relay node with minimal network latency in a scalable way. Though above works can find low-latency paths, they leave the routing problem to applications in overlay layer. In this paper, we trust that the low-latency routing paths should be provided by network core instead of applications.

VIII. Conclusion

This paper presented a method named SRUF to provide low-latency path routing service in wide area network. At its core, SRUF leverages the TIV phenomenons of the end-to-end latencies in the Internet to provide better alternative paths. We designed a novel method which has high scalability to pick up the correct indirect path for any pair of nodes. This method permits SRUF to absorb more ASes in an incremental way. Furthermore, SRUF leverages SRv6 to steer the flow along the selected path to promise that SRUF can be compatible with the legacy routing system. We have implemented SRUF using IPMininet, and evaluated it through both numerical experiments and emulation experiments. The experimental results with real-world datasets demonstrate that SRUF is a promising solution to reduce latency in wide area networks.

Acknowledgment

This work is partially supported by the National Key Research and Development Program of China (No. 2018YFE0207600 and No. 2020YFE0200500), the National Natural Science Foundation of China (No. U19B2024 and No. 61802421), European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie Grant (No. 824019), and the China Postdoctoral Science Foundation (No. 2019M663017)

References

[1] D. R. Mafioletti, A. B. Liberato, R. da Silva Villaça, C. K. Dominicini, M. Martinello, and M. R. N. Ribeiro, "Latency measurement as a virtualized network function using metherxis," *Computer Communication Review*, vol. 46, no. 4, pp. 14–16, 2016.

[2] B. Briscoe, A. Brunström, A. Petlund, D. A. Hayes, D. Ros, I. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing internet latency: A survey of techniques and their merits," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 3, pp. 2149–2196, 2016.

[3] C. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *In Proc. of ACM SIGCOMM*, 2013, pp. 15–26.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," in *In Proc. of ACM SIGCOMM*, 2013.

[5] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee, "Internet anycast: performance, problems, & potential," in *In Proc. of ACM SIGCOMM*, 2018.

[6] N. T. Spring, R. Mahajan, and T. E. Anderson, "The causes of path inflation," in *In Proc. of ACM SIGCOMM*, 2003.

[7] "Interconnection Test of Public Cloud," Available:https://mp.weixin.qq.com/s/rEO6ICeNvWIIBUIEHYewIg, 2020.

[8] G. Wang, B. Zhang, and T. S. E. Ng, "Towards network triangle inequality violation aware distributed systems," in *In Proc. of ACM IMC*, 2007.

[9] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee, "Triangle inequality variations in the internet," in *In Proc. of ACM IMC*, A. Feldmann and L. Mathy, Eds., 2009.

[10] C. Lumezanu, R. Baden, D. Levin, N. Spring, and B. Bhattacharjee, "Symbiotic relationships in internet routing overlays," in *In Proc. of USENIX NSDI*, 2009.

[11] D. A. Sontag, Y. Zhang, A. Phanishayee, D. G. Andersen, and D. R. Karger, "Scaling all-pairs overlay routing," in *In Proc. of ACM CoNEXT*, 2009.

[12] Y. Fu and E. Biersack, "MCR: structure-aware overlay-based latency-optimal greedy relay search," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3016–3029, 2017.

[13] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "An overlay architecture for high-quality voip streams," *IEEE Trans. Multimedia*, vol. 8, no. 6, pp. 1250–1262, 2006.

[14] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: a comprehensive survey of research activities, standardization efforts and implementation results," *CoRR*, vol. abs/1904.03471, 2019.

[15] "Network Latency Datasets," Available:https://github.com/uofa-rzhu3/NetLatency-Data, 2020.

[16] B. Zhang, R. A. Liu, D. Massey, and L. Zhang, "Collecting the internet as-level topology," *Computer Communication Review*, vol. 35, no. 1, pp. 53–61, 2004.

[17] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[18] N. McKeown, T. E. Anderson, H. Balakrishnan, G. M. Parulkar, L. L. Peterson, J. Rexford, S. Shenker, and J. S. Turner, "Openflow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[19] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Comput. Commun.*, vol. 67, pp. 1–10, 2015.

[20] "IPv6 Statistics," Available:https://https://www.internetsociety.org/deploy360/ipv6/statistics/, 2020.

[21] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. François, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *In Proc. of ACM SIGCOMM*, 2015.

[22] P. K. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall, "Improving the reliability of internet paths with one-hop source routing," in *In Proc. of USENIX OSDI*, 2004.

[23] "ROSE: Research on Open SRv6 Ecosystem," Available:https://netgroup.github.io/rose/, 2020.

[24] "IPMininet's documentation," Available:https://ipmininet.readthedocs.io/en/latest/, 2020.

[25] "Internet topology," Available:http://konect.cc/networks/topology/, 2020.

[26] "The Internet Topology Zoo," Available:http://www.topology-zoo.org/, 2020.

[27] M. S. Borella, S. Uludag, G. B. Brewster, and I. Sidhu, "Self-similarity of internet packet delay," in *In Proc. of IEEE ICC*, 1997.

[28] R. R. Sambasivan, D. Tran-Lam, A. Akella, and P. Steenkiste, "Bootstrapping evolvability for inter-domain routing with D-BGP," in *In Proc. of ACM SIGCOMM*, 2017.

[29] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. E. Anderson, "The end-to-end effects of internet path selection," in *In Proc. of ACM SIGCOMM*, 1999, pp. 289–299.

[30] "Performance-based BGP Routing Mechanism," Available:https://tools.ietf.org/id/draft-ietf-idr-performance-routing-02.html, 2020.

[31] "BGP Optimal Route Reflection (BGP-ORR)," Available:https://tools.ietf.org/html/draft-ietf-idr-bgp-optimal-route-reflection-20, 2020.

[32] T. Arnold, M. Calder, Í. Cunha, A. Gupta, H. V. Madhyastha, M. Schapira, and E. Katz-Bassett, "Beating BGP is harder than we thought," in *In Proc. of Hotnet*, 2019.

[33] M. Calder, R. Gao, M. Schröder, R. Stewart, J. Padhye, R. Mahajan, G. Ananthanarayanan, and E. Katz-Bassett, "Odin: Microsoft's scalable fault-tolerant CDN measurement system," in *In Proc. of USENIX NSDI*, 2018.

[34] K. Yap, M. Motiwala, J. Rahe, S. Padgett, and et al, "Taking the edge off with espresso: Scale, reliability and programmability for global internet peering," in *In Proc. of ACM SIGCOMM*, 2017.