

Efficient Slice-Parallel Distributed Probing in SRv6 Networks

Kaiyang Zhao¹, Han Zhang¹, *Member, IEEE*, Yao Tong¹, Yahui Li¹, Xingang Shi¹, *Member, IEEE*, Zhiliang Wang¹, *Member, IEEE*, Xia Yin¹, *Senior Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

Abstract—Segment Routing over IPv6 (SRv6) is widely deployed, where operators construct numerous parallel SR-based network slices. An accurate diagnosis of latency bottlenecks in SRv6 tunnels is essential to maintain service-level objectives. However, building a distributed, at-scale probing system is non-trivial: SRv6 priority policies, SR-aware multipath forwarding, and slice isolation collectively invalidate assumptions made by existing diagnostic methods. In this paper, we present **SRmesh**, a distributed system for diagnosing latency bottleneck links in SRv6 networks. First, we adopt distributed SR-based probing agents to control routing paths and ensure that probe packets emulate per-slice production traffic. Second, we employ a latency-based multipath inference that runs in parallel across slices to resolve SR-induced routing ambiguities. Third, we introduce a slice-parallel progressive diagnosis that incrementally reuses probe results to reduce redundant measurements, optimizing diagnostic overhead for large-scale SRv6 overlay networks. We implement a prototype of **SRmesh** and conduct extensive evaluations on 247 real network topologies. The results indicate that **SRmesh** achieves high diagnostic accuracy with a 93.4% reduction in probe overhead, demonstrating its practicality and scalability in large-scale SRv6 environments.

Index Terms—Segment routing, IPv6, latency bottleneck, network slicing.

I. INTRODUCTION

SEGMENT Routing (SR) has been widely adopted in various networks [1], such as Software-Defined Wide Area Networks (SD-WANs) that interconnect distributed sites. They require fine-grained traffic engineering (TE) [2] and use SR over IPv6 (SRv6) to deliver multiple, parallel network slices,

each configured to meet specific Service Level Objectives (SLOs), including latency demands. For example, latency-sensitive slices are provided for services such as remote healthcare [3], electronic financing [4], and vehicular communications [5]. Operators must identify and rectify bottlenecks in these parallel slices if the end-to-end latency exceeds established thresholds.

Diagnosing latency bottlenecks in SRv6 networks remains challenging due to several problems. First, it is difficult to accurately reflect the behavior of the actual service traffic. Measurement methods must avoid generating misleading results that may result from packet prioritization and firewall filtering. These phenomena are common in SRv6 networks supporting multiple parallel slices. Second, multipath routing from SR complicates the diagnostic process. The algorithm must be able to locate the bottleneck link among multiple potential paths with precision. Third, the solution must maintain low overhead to remain practical in large-scale SRv6 overlay networks.

Existing approaches fall into two categories, passive and active measurement. Passive approaches [6], [7], [8], [9], [10], [11], [12], [13], [14] cannot provide early warnings for links that may experience future latency anomalies such as backup or rerouted paths during failures, and they also cannot offer reliable performance estimates required for dynamic SR TE optimization. Active approaches can be further classified into in-network and host-based approaches according to the necessity of modifications to network devices. In-network approaches [15], [16], [17] require changes to all forwarding devices, which makes them impractical for SR deployments aiming to support incremental implementation. One motivation for adopting SR is to avoid modifications to existing infrastructure. Host-based approaches [18], [19], [20], [21], [22], [23], [24] usually incur significant probing overhead, and the probe packets may not follow the same forwarding paths as actual service traffic. In addition, multipath routing in SR introduces further complexity in bottleneck diagnosis.

Vision. We aim to develop a lightweight, accurate, and fast *distributed probing system* for locating link-level latency bottlenecks in SRv6 networks. Similar to BFD [25], [26], a primary application of such a system is to enable rerouting decisions once a bottleneck is identified. In addition, the latency measurements collected by distributed probing agents during the process can serve as precise inputs for rerouting orchestration, thereby improving the accuracy of TE decisions.

A latency bottleneck link is defined as a link whose performance violates predefined latency objectives. In a properly

Received 12 January 2026; revised 10 April 2026; accepted 12 May 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor H. Shen. Date of publication 21 May 2026; date of current version 10 June 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62572269 and Grant 62394322. Part of this work was presented at the 33rd IEEE International Conference on Network Protocols (ICNP 2025) [DOI: 10.1109/ICNP65844.2025.11192455]. (*Corresponding author: Han Zhang.*)

Kaiyang Zhao and Yao Tong are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.

Han Zhang, Xingang Shi, and Zhiliang Wang are with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100190, China, also with Zhongguancun Laboratory, Beijing 100081, China, and also with the Tsinghua National Laboratory for Information Science and Technology (TNLIST), Beijing 100084, China (e-mail: zhhan@tsinghua.edu.cn).

Yahui Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100190, China.

Xia Yin and Jianping Wu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, also with Zhongguancun Laboratory, Beijing 100081, China, and also with the Tsinghua National Laboratory for Information Science and Technology (TNLIST), Beijing 100084, China.

Digital Object Identifier 10.1109/TON.2026.3695506

2998-4157 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15, 2026 at 12:24:19 UTC from IEEE Xplore. Restrictions apply.

functioning network, the latency of each link remains stable over time. These latency objectives can be obtained from empirical measurements by multiplying periodic latency values with a redundancy factor or can be statically configured by network operators. The scope of this work does not include root cause analysis or remediation mechanisms.

SRmesh. We propose SRmesh, a host-based distributed probing system specifically designed for SRv6 networks to diagnose link-level latency bottlenecks in parallel slices. In contrast to existing approaches, SRmesh achieves precise localization of problematic links while maintaining low overhead. To achieve these goals, SRmesh facilitates both efficient distributed probing and accurate diagnosis with the following **novel designs**:

First, we design the **distributed agent** for precise and impartial latency measurement in SRv6 (Section III-A). To eliminate path asymmetry, we embed forward and reverse SR paths in the probe packets. Furthermore, we improve TCP-based probing, accelerated by the extended Berkeley Packet Filter (eBPF) [27], to reduce overhead while preserving the behavioral fidelity of actual service traffic.

Second, we propose **multipath inference** to estimate Equal Cost Multipath (ECMP) routing selections in SR by minimizing discrepancies from measurements collected via distributed probing (Section III-B). The design transforms routing uncertainty into a constraint system and, using a greedy algorithm together with a specialized error-resolution solver, infers the path selections of the actual probe packets issued by distributed agents.

Lastly, we introduce the **progressive diagnosis** to reduce probing overhead without dependence on particular topological configurations (Section III-C). Our diagnosis incrementally reutilizes the results of existing probes to infer unmeasured paths, achieving efficient coverage through a progressive strategy.

Our **contributions** are summarized as follows:

- We conduct a wide study of the new requirements imposed by latency bottleneck link diagnosis in SRv6 networks (Section II).
- We design and implement SRmesh, a scalable and accurate diagnosis framework for SRv6 networks. It integrates distributed probing, multipath inference, and progressive diagnosis to localize bottleneck links with minimal overhead (Section III).
- We perform a comprehensive evaluation of SRmesh, demonstrating its effectiveness in accurately identifying bottleneck links. Experiments on real topologies show that SRmesh achieves high diagnostic accuracy while reducing probe overhead by 93.4%. To further validate the practical viability of SRmesh, we deploy a prototype in a controlled testbed environment (Section IV).

This paper serves as an extended version of a prior work published in the proceedings of the 33rd IEEE International Conference on Network Protocols [28]. Compared to the previous work, this version includes (1) several new sections about distributed system deployment, diagnostic-related error resolution algorithms, and parallel processing, and (2) new evaluation results using a comprehensive set of 247 real topologies. Other sections are heavily revised with new content or have been deleted. We open-source our SRmesh prototype at <https://github.com/DeepShield-AI/SRmesh>

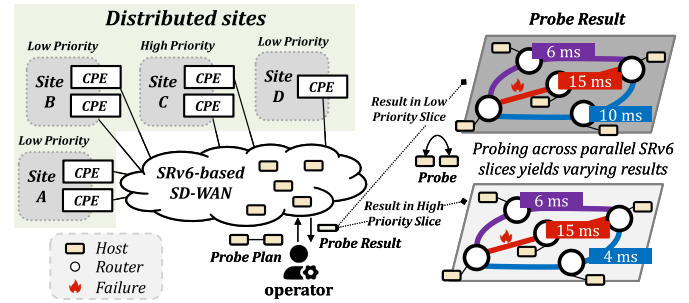


Fig. 1. Diagnosing in SRv6-based SD-WAN. Sites connect through Customer Premises Equipment (CPE) devices, which assign traffic to parallel slices. Probing only low-priority slices yields erroneous results.

II. BACKGROUND AND MOTIVATION

A. Operator Concerns in SRv6 Networks

SR is a source routing paradigm that encodes paths or functions as segments contained within the Segment Routing Header (SRH) of a packet. In SRv6, each Segment Identifier (SID) is a 128-bit address that can incorporate a slice identifier, thereby enabling the SID to represent both forwarding behavior and slice semantics. The ingress node appends the SRH, and packets are then forwarded according to the segment list, with each intermediate node processing the active segment as the destination address.

SRv6 can be employed to construct parallel network slices, where each slice corresponds to a logically isolated service plane with specific performance or policy requirements. By assigning distinct SIDs and segment lists to individual slices, SRv6 enables traffic differentiation and resource allocation within a shared physical infrastructure.

The architectural simplicity of SRv6 makes it well suited to dynamic networking scenarios such as SD-WAN [1]. Nevertheless, such environments are susceptible to gray failures that impair performance by inducing abnormal latency while avoiding total service disruption [29].

B. New Requirements for Diagnosis in SRv6 Networks

Requirement 1: Efficient distributed probing agent that generates deterministic probe packets using SR. In current SRv6 networks, probe packets are generated primarily using ICMP-based or UDP-based techniques [30], [31], [32], including SR Ping and SR Traceroute. Nevertheless, packets from certain protocols may have lower priority in the network, which can lead to misleading results. Second, specific network devices, such as firewalls, may restrict certain types of traffic, resulting in probe failures. In network environments where TCP traffic dominates and is prioritized, TCP-based probing tools should be used. In addition, these tools should be considered complementary rather than substitutive.

In addition, SRv6-based slicing can further aggravate this problem. As illustrated in Fig. 1, a router failure may initiate rerouting based on probing traffic originating from a low-priority slice, which reports a 6 ms path, while the optimal 4 ms path for the high-priority slice is overlooked. This discrepancy results in inefficient routing decisions and performance degradation. To mitigate such suboptimal outcomes, SR measurement mechanisms must accurately reflect the behavior

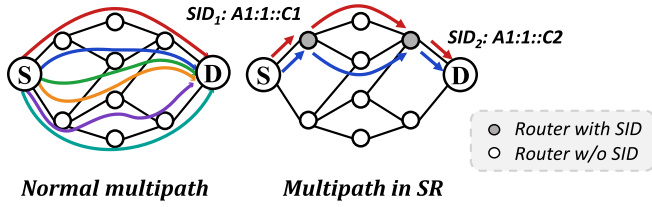


Fig. 2. Multipath diversity in normal routing vs. SR. SID-related constraints limit path options.

of actual traffic flows to ensure that path selection aligns with service priorities.

Challenge for R1: Deterministic operation requires the completion of slice-level probing and the inclusion of the corresponding SRH in each packet. In addition, the probing mechanism demands explicit control of both the forward and reverse transmission paths by means of SRHs. This operation results in additional header overhead, which heightens the probability of exceeding the Maximum Transmission Unit (MTU) limit and may consequently lead to packet fragmentation or reduced transmission efficiency.

Requirement 2: Fast and accurate diagnosis from inferring distributed probing results. In SRv6 networks, multipath routing introduces uncertainty in the routing trajectories of probe packets. The precise path traversed by each probe is indeterminate, thereby complicating the association of observed performance metrics with specific links. As illustrated in Fig. 2, SR-based multipath differs fundamentally from conventional multipath routing. In SR, the source node explicitly specifies a sequence of intermediate nodes that the packet must traverse, enabling fine-grained control over the forwarding path. However, multiple alternative paths may still exist between any two segments within the segment list, and this multipath behavior arises only among nodes possessing SIDs. This intrinsic characteristic provides additional topological and structural information that can be leveraged to enhance the diagnosis and resolution of multipath-related performance issues.

Moreover, SR paths are inherently unidirectional, and when bidirectional lists are established [33], the presence of multipath between SIDs can introduce path asymmetry. This asymmetry invalidates the common assumption that one-way latency equals half of the Round-Trip Time (RTT), thereby resulting in non-deterministic path measurement and reduced accuracy in latency estimation.

Challenge for R2: SR introduces path variability due to ECMP among SIDs. Consequently, diagnostic analysis must consider all feasible path combinations, rendering the problem NP-hard since the number of possible paths increases exponentially with network scale. Hence, it becomes crucial to design efficient polynomial-time approximation algorithms capable of accurately capturing network behavior while maintaining computational tractability.

Requirement 3: Cost-effective probing in large-scale SRv6 networks. In traditional networks, each router is configured with a distinct IP address. By contrast, SR-based overlay networks adopt a two-dimensional addressing scheme. In this scheme, the address of a physical node together with a slice ID identifies a logical node inside a slice. In large-scale SRv6 deployments that may comprise hundreds of physical nodes, a single router can belong to thousands of slices.

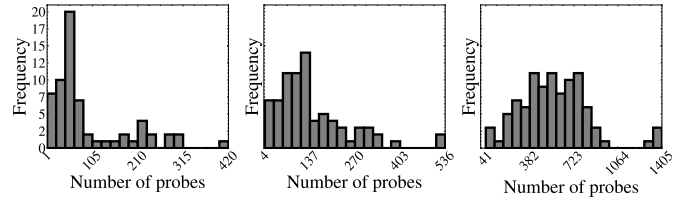


Fig. 3. Distributed probing redundancy under full-mesh measurements in BellCanada, UsSignal, and VtWavenet topologies. For each pair, paths follow shortest path routing, and the amount of redundancy is obtained through graph theoretic analysis.

TABLE I
A SUMMARY OF EXISTING APPROACHES

Category		Approaches	Requirements ²		
			R1	R2	R3
In-network		NetVision [15], V-Trace [16], Zoonet [17]	●	●	●
Host-based	General	Pingmesh [18], VNET-pingmesh [21]	○	○	○
	DCN-specific ¹	RD-probe [23]	●	●	●
	SR-specific	SCMon [19]	○	●	●
		SRmesh	●	●	●

¹ The approaches exploit the particular nature of data center networks.

² ● indicates that the requirement is fully satisfied, ● indicates partial satisfaction, and ○ indicates that the requirement is not satisfied.

The probing overhead can be illustrated by an example. Consider an overlay network that contains 100 physical routers and 4,096 slices. Under these conditions, Pingmesh [18] generates approximately 5,000 probing paths in the physical network, while the overlay network requires over 20 million probing paths, resulting in an increase $O(10^3)$.

However, a large proportion of the probes are redundant. As shown in Fig. 3, under a full-mesh probing scheme, many links are measured repeatedly, with links probed more than 1,400 times. This redundancy becomes increasingly severe in large-scale topologies and provides little diagnostic value while it consumes significant resources.

Challenge for R3: In structured networks exemplified by data center networks (DCNs), the topology shows regular patterns that support efficient probing strategies. In contrast, SRv6 networks do not exhibit fixed structural regularity. The challenge is to design scalable algorithms that, using a limited number of probes, can cover all slices and their SR paths while still achieving accurate localization of performance bottlenecks.

C. Motivating SRmesh

We divide existing probing approaches into **host-based probing** and **in-network probing**, depending on whether modifications to network devices are required. A summary of this categorization is presented in Table I.

In-network probing incurs significant overhead and lacks scalability in SRv6 networks. NetVision [15] attaches two separate label stacks (SR stack and INT stack), which increases additional overhead. As probes traverse the network, SR labels are popped to guide forwarding and INT labels are pushed to record telemetry information. V-Trace [16] installs measurement rules on virtual forwarding devices to trace packets with fine granularity. Because rules must be

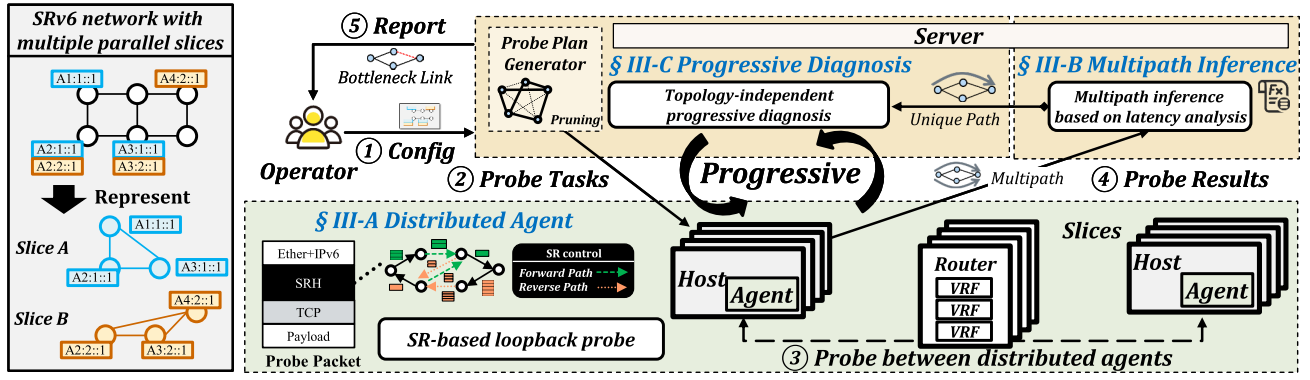


Fig. 4. Overview of SRmesh.

configured on every forwarding device, sustained deployment reduces forwarding performance, with an average degradation of about 6% per rule. Zoonet [17] enables tracing at each hop and supports probe pruning, but similarly requires modifying every device in the forwarding path, which makes adoption difficult in SRv6 networks operated by ISPs.

Host-based distributed probing lacks determinism and struggles with multipath ambiguity. Pingmesh [18] and VNET-pingmesh [21] perform end-to-end full-mesh probing, but introduce significant overhead and are sensitive to multipath effects. To mitigate these challenges, RD-Probe [23], based on IP-in-IP [34], combines randomized and deterministic strategies to balance path coverage and measurement cost. The key limitation lies in the fact that the RD-Probe is designed primarily for DCNs. Similarly to a range of active probing approaches [20], [22], [24] developed for packet loss detection in DCNs, it cannot be easily extended to general SR networks. DCNs have structured topologies that facilitate the design of probing schemes. However, SRv6 networks use topologies that cannot be predicted in advance, making the design process more difficult. SCMon [19] targets SR networks and mitigates multipath ambiguity by adjusting IGP weights within the monitoring plane. Nevertheless, it does not support slice-level monitoring, as the monitoring plane cannot differentiate between SR slices. Meanwhile, probe packets, typically ICMP, are deprioritized in the network, making them susceptible to misleading latency or loss measurements. Such probes cannot guarantee stable traversal paths under multipath routing, leading to inconsistent results.

III. SRMESH FRAMEWORK

We illustrate the overall architecture of the SRmesh in Fig. 4. SRmesh has two main components: distributed **agent** and **server**. The **server** obtains network configuration information from the network operator via an interface. The inputs include the physical topology and the SR configuration that specifies the slice-related information (1).

The SR configuration defines a series of overlay networks, where each overlay is associated with one slice. These overlays induce logical paths on the shared physical infrastructure. As shown in Fig. 4, different slices use different SRv6 SID ranges (e.g., A1:1::C1 for Slice A and A1:2::C1 for Slice B), although they traverse the same physical nodes.

Based on the configured slice-specific SIDs, the controller derives a set of slice-sensitive forwarding paths and distributes

the corresponding probing plans to the **agents** (2). SRmesh ensures that probes follow the same SID lists as actual service traffic, preserving the slice-level path semantics during performance measurement.

The **agent** is instantiated on selected container nodes, virtual machines, or physical servers. It uses predefined eBPF hooks together with probing extensions to collect probing data (3), which is then delivered to the **server** (4). The interaction from (2) to (4) progresses in a progressive manner until the diagnosis is completed. The system finally reports the bottleneck links and, for each link, provides its logical representation within the SRv6 slice or policy together with the corresponding physical path in the underlying topology (5).

A. Distributed Agent

Distributed agents in SRv6 networks are responsible for performing deterministic probing, which is required for an accurate diagnosis of bottleneck links. For these agents, determinism means removing uncertainty on both forward and reverse paths so that asymmetry-induced bias is avoided and the forwarding treatment of production traffic is matched, without perturbation from priority-based scheduling. Traditional tools such as ping do not satisfy these conditions, because their probe packets can be filtered or handled with lower forwarding priority and therefore fail to reflect the behavior of in-tunnel business traffic, in particular TCP-based services.

★ **DESIGN 1: SR-based loopback probe.** In response, SRmesh adopts a TCP-based active probing method for the measurement of SR tunnel quality, which must overcome several main challenges. One concern is path asymmetry, which may cause measurement errors. The second issue is scalability. The baseline method must establish thousands of TCP sessions per second, which risks exhausting resources.

Probe packet. In SRv6 networks, routing from a source SID to a destination SID usually depends on ECMP hash functions that operate on the five tuple. This behavior often produces different forward and reverse paths. To mitigate this issue, we adopt a deterministic probing mechanism. At the sending node, the SR list is derived from the SR configuration and placed in the packet payload together with a timestamp and the SR list for the return direction. When the probe reaches the receiving node, and before it enters the SRv6 processing pipeline, the agent reads the return SR list from the payload

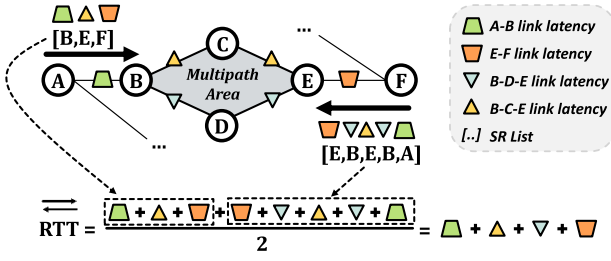


Fig. 5. **SR-based loopback probe.** Reverse SR lists are embedded to enable reverse path traversal, yielding RTT under multipath.

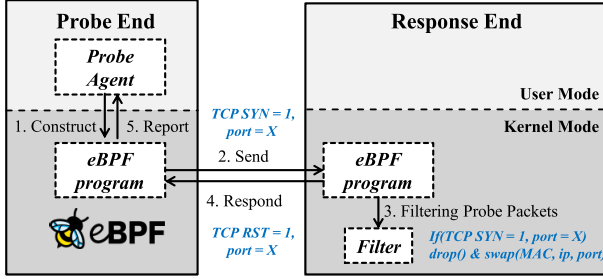


Fig. 6. **TCP Acceleration.** eBPF bypasses TCP kernel handling to accelerate RTT measurement.

and builds a new SRH that explicitly specifies the reverse source route. The sender then computes the RTT using the timestamps. If the ingress router has a SR policy configured, the host inherits the corresponding SR list and embeds it into the probe. Otherwise, the host constructs a minimal SR list from the network topology and configuration.

SRH control. The SR list contains entries that describe possible reverse paths for multiple SID pairs. For example, when several paths exist between nodes B and E, a reverse SR list from B to E is placed between E and B, as illustrated in Fig. 5. At the receiving node, before further processing, the SR list carried in the payload is written into the packet. In this way, the packet traverses the multiple SID paths and the measured latency corresponds to the cumulative latency along a sequence of predetermined paths. When the reverse SR list is omitted from the header of the forward probe, the responder agent inserts it on the return path so that the overall length of the SR list is reduced by half.

TCP acceleration. The agent applies eBPF to attach to kernel-level TCP state transitions and thereby mitigates the inefficiencies of traditional TCP probing, as illustrated in Fig. 6. When TCP SYN packets are intercepted before entering the protocol stack, SRmesh produces and returns RST responses locally. In this way, RTT measurement is accelerated without establishing complete TCP connections, and the overhead associated with traditional TCP connection establishment is reduced. Compared with implementations in the user space, eBPF supports packet processing with lower latency and reduced CPU utilization. In networks that contain up to thousands of slices, this acceleration is necessary to maintain responsiveness.

Distributed system deployment. The probe data uses distributed storage, remaining only at the probe source to avoid data synchronization issues. It is periodically uploaded to a central server and released upon confirmation to reduce

the burden on the distributed agent. A maximum number of storage entries should be set during coding to prevent exhausting device storage in case of server connection failures.

SRmesh supports partial deployment, where only a subset of routers hosts probing servers. To preserve deterministic probing, SRmesh introduces a springboard-router mechanism. This mechanism logically equips every router with probing capability. For a router without a local server, a springboard router sends two probes for each measurement. One is a synthesized probe that reproduces the target router's intended probing behavior and carries an appended SR round-trip segment to the target router. The other is a baseline SR round-trip probe between the springboard router and the target router. SRmesh subtracts the baseline metric from the synthesized metric to recover the target router's measurement result. This overhead is limited, as each measurement adds only one extra probe and one SR round-trip traversal between the springboard router and the target router.

B. Multipath Inference

The multipath makes the attribution of end-to-end latency uncertain because a single measured latency value can arise from any one of several internal paths. The resulting constraint system for these latency observations therefore contains deterministic constraints, such as algebraic equalities, together with uncertain constraints, such as disjunctive expressions. Solving this system is computationally intractable, because the number of possible path combinations grows exponentially with the size of the network, which makes the problem NP-hard. The model is specified as follows.

$$\min \sum_{i=1}^m \left| \sum_{j=1}^n a_{ij} \cdot x_j + \sum_{s=1}^{S_i} \sum_{j=1}^n y_{ijs} \cdot x_j - d_i \right| \quad (1)$$

Subject to:

$$\sum_{j=1}^n y_{ijs} = 1, \quad \forall i \in \{1, \dots, m\}, \quad \forall s \in \{1, \dots, S_i\} \quad (2)$$

$$x_j \geq 0, \quad \forall j \in \{1, \dots, n\} \quad (3)$$

$$y_{ijs} \in \{0, 1\}, \quad \forall i \in \{1, \dots, m\}, \quad j \in \{1, \dots, n\}, \quad s \in \{1, \dots, S_i\} \quad (4)$$

The objective in (1) minimizes the sum of the absolute deviation between the predicted and observed end-to-end delays d_i , where x_j denotes the delay of the link j , and y_{ijs} selects the uncertain link j in the segment s for the path i , which contains S_i multipath segments. For each given path i , the predicted delay is expressed as the sum of two components, namely the delays on known links j with coefficients a_{ij} and uncertain links. Constraint (2) enforces that exactly one uncertain path is chosen in every segment, with y_{ijs} indicating valid candidates. Constraint (3) ensures that delays are non-negative, and constraint (4) sets y as binary.

Existing approaches to handling multipath rely on extensive probing. For example, the value of the Flow Label [35] field in the IPv6 header can be varied to trigger different ECMP paths. Under this strategy and using coupon collector theory, all links are expected to be covered. In practice, the system may transmit thousands of packets, each with a different ECMP hash value, and assume that this volume achieves complete

Algorithm 1 Greedy Inference

Input: C_d, C_u, E
Output: $\mathcal{S}$: Estimated values to unknown latencies

```

1:  $\mathcal{S} \leftarrow \emptyset$ 
2: for each  $c \in C_d$  do
3:   Solve  $c$  by elimination, update  $\mathcal{S}$ 
4: end for
5: while  $\mathcal{S}$  is updated do
6:   for each  $c_u \in C_u$  not fully resolved do
7:      $\mathcal{V}_c \leftarrow \{\kappa_i + \alpha_i \mid \forall \kappa_i \in (\alpha_i, \kappa_i) \in c_u\}$ 
8:     Partition  $\mathcal{V}_c$  into  $\{\mathcal{G}_1, \mathcal{G}_2, \dots\}$  where  $\forall v_p, v_q \in \mathcal{G}_k$ ,
        $|v_p - v_q| \leq E$ 
9:     for each  $\mathcal{G}_k$  do
10:      Randomly select representative  $v_k^* \in \mathcal{G}_k$ 
11:      Tentatively assign  $\kappa_i \leftarrow v_k^*$  for  $(\alpha_i, \kappa_i) \in \mathcal{G}_k$ 
12:      Compute induced error  $E_k$ 
13:    end for
14:    if  $\exists \mathcal{G}_k$  with acceptable  $E_k$  then
15:      Choose  $\mathcal{G}^* = \arg \min_k E_k$ 
16:      Update  $\mathcal{S}$  with assignments from  $\mathcal{G}^*$ 
17:    end if
18:  end for
19: end while
20: for each unresolved  $c_u \in C_u$  do
21:   Compute mean  $\bar{v} \leftarrow \frac{1}{|\mathcal{V}_c|} \sum_{v_j \in \mathcal{V}_c} v_j$ 
22:   Assign  $d_j \leftarrow \bar{v}$  for all  $(\alpha_i, d_j) \in c_u$  and Update  $\mathcal{S}$ 
23: end for
24: return  $\mathcal{S}$ 

```

path coverage. However, this cost-insensitive probing increases the number of probe packets.

★ **DESIGN 2: Multipath inference based on latency analysis.** In multipath routing scenarios, SRmesh applies an iterative greedy procedure to support diagnosis, as summarized in Algorithm 1. This procedure infers routing paths by interpreting end-to-end latency measurements while enforcing the constraints.

The inference procedure first processes deterministic constraints C_d through algebraic elimination, directly obtaining latency values from exact constraints (lines 2 to 4). Each uncertain constraint c_u is expressed as a collection of pairs (α, κ) , where α denotes a known latency and κ denotes an unknown latency to be estimated. For each constraint c , the set $\mathcal{V}_c$ gathers these unknown values for further processing (line 7). Following deterministic resolution, the algorithm evaluates the candidate latency values for each remaining disjunctive constraint C_u and organizes them into clusters $\mathcal{G}$ according to a proximity threshold E (line 8). Within each cluster, a representative value v_k^* is selected (line 10) and tentatively assigned (line 11), while the induced error E_k is calculated for each assignment (line 12). The assignment with the least permissible error is utilized to revise the solution set (lines 14 to 16). This loop proceeds iteratively because newly resolved values can simplify additional constraints and enable further updates. If certain disjunctive constraints remain unresolved after the greedy inference step, the algorithm computes the mean of the candidate values in each group and assigns this mean as an approximation (lines 20 to 22). The procedure continues until no further updates are possible.

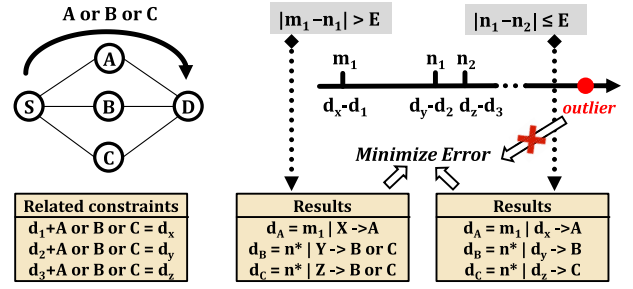


Fig. 7. Example of multipath inference. Candidate latencies are grouped by proximity, while the impact of outliers is handled by robust error evaluation.

To make the procedure more explicit, consider the example in Fig. 7. In the presence of disjunctive constraints such as $d_1 + (d_a \vee d_b \vee d_c) = b$, the algorithm examines the differences between candidate latency values, such as $|d_a - d_b|$ and $|d_b - d_c|$. If the absolute difference between any pair of candidates is larger than the threshold E , the corresponding values are interpreted as belonging to different route paths. If the difference remains within this threshold, the corresponding values (e.g., d_b and d_c) are grouped and a representative value is selected to approximate them. The algorithm greedily proceeds to assign values that reduce the overall approximation error. In a corner case, a latency increase on one link may be offset by a latency decrease on another. In practice, this risk is mitigated by the multipath design, since a faulty link typically appears in multiple probe paths.

Calculation of error. During deployment, probes occasionally fail or exhibit abnormal latency. This poses a significant challenge to the SRmesh, as these anomalous values can lead to diagnostic errors, particularly in error calculation. Therefore, we propose a novel error calculation method.

For a tentative assignment generated from cluster $\mathcal{G}_k$ on an unresolved constraint c , let r_j denote the induced residuals after substituting the current estimates into the end-to-end latency equations. We estimate a robust scale $\hat{\sigma}_c = \text{MAD}(r)$ and form standardized residuals $\tilde{r}_j = r_j / \hat{\sigma}_c$, so that the data-fit term has bounded influence:

$$E_k^{\text{fit}} = \sum_{j \in c} \rho_\delta(\tilde{r}_j), \quad \rho_\delta(x) = \begin{cases} \frac{1}{2}x^2, & |x| \leq \delta, \\ \delta\left(|x| - \frac{1}{2}\delta\right), & |x| > \delta, \end{cases} \quad (5)$$

which preserves quadratic sensitivity near zero and limits the leverage of large deviations. To avoid spreading a true bottleneck across many links, we penalize latency increments $\Delta d_i = d_i - d_i^{(0)}$ relative to a baseline via a sparsity prior (optionally grouped by paths $\mathcal{P}$):

$$E_k^{\text{sp}} = \lambda \sum_i w_i |\Delta d_i| + \lambda_g \sum_{g \in \mathcal{P}} \|\Delta d_g\|_2, \quad (6)$$

where w_i compensate for heterogeneous scales and $\|\Delta d_g\|_2$ concentrates changes on few routes. The composite score used at Algorithm 1 (line 12) is

$$E_k = E_k^{\text{fit}} + E_k^{\text{sp}}, \quad (7)$$

and the greedy update accepts the feasible assignment with minimal E_k . If a constraint remains unresolved, its unknown

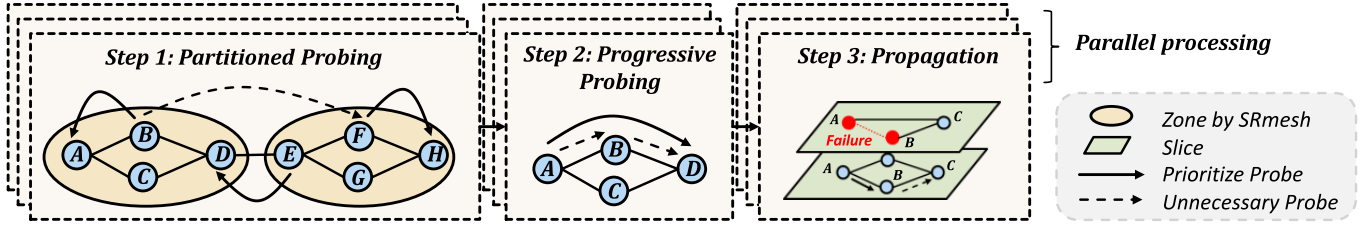


Fig. 8. Workflow of topology-independent progressive diagnosis.

segment is filled by the robust aggregator that minimizes the weighted absolute deviation over its candidate set $\mathcal{V}_c$,

$$m^* = \arg \min_m \sum_{v_j \in \mathcal{V}_c} w_j |v_j - m|, \quad (8)$$

i.e., the weighted median, which yields an estimate less sensitive to outlying candidates and consistent with the above objective.

C. Progressive Diagnosis

SRmesh aims to maintain complete path coverage across arbitrary topologies while lowering the required number of probes. A straightforward baseline performs full-mesh probing inside each slice, which leads to substantial resource consumption. In contrast, the progressive diagnosis in SRmesh adopts a partitioned and staged probing strategy that preserves coverage and at the same time reduces overhead. A central challenge is to keep this efficiency under diverse topologies, because structural differences among networks influence how effective a given probe planning scheme can be.

★ **DESIGN 3: Topology-independent progressive diagnosis.** First, SRmesh employs a *divide-and-conquer* strategy that partitions the network into sub-networks with dense connections inside each partition. The algorithm remains independent of the specific topology and does not rely on any particular structural pattern. Second, *progressive probing* is applied; once a long path has been verified, shorter paths inside the same partition can be inferred, which reduces redundant probes. In addition, *cross-slice links* are scheduled with high priority in early probing so that failures that may influence multiple slices can be detected quickly, as shown in Fig. 8.

To formalize the problem, we introduce a model for allocating probes among multiple network slices, with emphasis on coverage. The objective is to lower probing overhead while still capturing inter-slice dependencies and ensuring that all relevant paths are exercised. The model is specified as follows.

$$\min \sum_{i=1}^N \sum_{u,v \in V_{s_i}} X_{s_i}(u, v) \quad (9)$$

Subject to:

$$Y_{s_i}(u, v) \geq 1, \quad \forall (u, v) \in E_{s_i} \quad (10)$$

$$Y_{s_i}(u, v) = \sum_{u,v \in V_{s_i}} X_{s_i}(u, v) \cdot A_{s_i}(u, v) \quad (11)$$

$$X_{s_i}(u, v) \leq A_{s_i}(u, v), \quad \forall u, v \in V_{s_i} \quad (12)$$

$$X_{s_i}(u, v) + X_{s_i}(v, u) \leq 1, \quad \forall u, v \in V_{s_i} \quad (13)$$

$$X_{s_i}(u, v) \in \{0, 1\}, \quad \forall u, v \in V_{s_i} \quad (14)$$

Algorithm 2 Partitioned Probe

Input: $G = (V, E)$

Output: Partition set $\mathcal{Z}$

```

1:  $\mathcal{Z} \leftarrow \{\{v\} \mid v \in V\}$ 
2:  $m \leftarrow |E|$ 
3: while  $\mathcal{Z}$  updated do
4:   for each  $v \in V$  do
5:     Compute  $\Delta Q(v, \mathcal{Z})$  for all adjacent partitions  $\mathcal{Z}$ 
6:      $Z^* \leftarrow \arg \max_{\mathcal{Z}} \Delta Q(v, \mathcal{Z})$ 
7:     if  $\Delta Q(v, Z^*) > 0$  then
8:       Move  $v$  to  $Z^*$ 
9:       update  $\mathcal{Z}$ 
10:    end if
11:  end for
12: end while
13: return  $\mathcal{Z}$ 
```

First, objective (9) minimizes the total size of the probing pair set across all slices, where the binary decision variable $X_{s_i}(u, v)$ equals 1 if nodes u and v form a probing pair in slice s_i , and 0 otherwise. As a fundamental constraint, full coverage (10) requires that each edge in slice s_i be covered by at least one probing pair. Based on this, definition (11) states that the coverage of each edge is determined by the total number of probing pairs, with $A_{s_i}(u, v)$ indicating the presence of a routing path between nodes u and v . Constraint (12) enforces that probing pairs are created only for node pairs that have an available routing path. From the point of view of efficiency, the probing pair symmetry constraint (13) limits the selection to at most one direction for each unordered node pair, which prevents redundant probing. Finally, the binary constraint (14) ensures that each probing pair is selected or not selected, without intermediate values.

Partitioned probing. Each slice is decomposed into sub-networks that exhibit denser connectivity inside partitions, whereas links between partitions are heavily reused by end-to-end full-mesh probing. This decomposition enables more efficient probing by focusing first on local connections, which in turn reduces the total number of probes required to achieve complete coverage.

Algorithm 2 describes the modularity maximization partitioning method. At the beginning, each node is assigned to its own partition (line 1). The algorithm then improves the modularity value Q defined by Equation (15) by repeatedly applying a greedy node relocation strategy. In every iteration, the modularity increment ΔQ is evaluated for each node with respect to neighboring partitions. A node v is then moved into the partition P^* that provides the maximal positive modularity

Algorithm 3 Progressive Probe

Input: Path set $\mathcal{P}$
Output: Abnormal set $\mathcal{A}$

```

1:  $\mathcal{A} \leftarrow \emptyset, \mathcal{N} \leftarrow \emptyset, U \leftarrow \mathcal{P}$ 
2: while  $U \neq \emptyset$  do
3:    $S \leftarrow \emptyset$ 
4:   for each  $p \in U$  do
5:     if  $\nexists q \in \mathcal{N}$  such that  $\text{path}(p) \subseteq \text{path}(q)$  then
6:        $S \leftarrow S \cup \{p\}$ 
7:     end if
8:   end for
9:   Probe all  $p \in S$ 
10:   $\mathcal{A} \leftarrow \mathcal{A} \cup \{p \in S \mid \text{probe}(p) = \text{abnormal}\}$ 
11:   $\mathcal{N} \leftarrow \mathcal{N} \cup \{p \in S \mid \text{probe}(p) = \text{normal}\}$ 
12:   $U \leftarrow U \setminus S$ 
13: end while
14: return  $\mathcal{A}$ 

```

increase (lines 7 to 9). This refinement step is repeated until no node can obtain a positive modularity improvement, which marks convergence (line 3). After convergence, connectivity inside each partition is verified by issuing full mesh probes within that partition. Connectivity between partitions is checked by choosing representative nodes, for example nodes D and E in Fig. 8, from each partition and including them in a second stage full mesh probing phase.

The modularity Q that steers the partition optimization is given by:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \quad (15)$$

where A_{ij} denotes the adjacency matrix, k_i and k_j represent the degrees of nodes i and j , respectively, m is the total number of edges, c_i denotes the partition assignment of node i , and $\delta(c_i, c_j)$ equals 1 if nodes i and j belong to the same partition and 0 otherwise.

Progressive probing. SRmesh is based on the premise that if a long probe path exhibits normal behavior, then every shorter path contained in that probe path also behaves normally. This assumption promotes a targeted probing approach, enabling a single long probe path (e.g., A-D in Fig. 8) to efficiently infer the status of multiple shorter paths (e.g., A-B and B-D in Fig. 8). In this way, the total number of probes can be reduced substantially while maintaining diagnostic accuracy.

Algorithm 3 specifies the progressive probing strategy in formal terms. The procedure starts by setting both the abnormal path set $\mathcal{A}$ and the normal path set $\mathcal{N}$ as empty and considers all paths in $\mathcal{U}$ as initially pending. During each iteration, it constructs a candidate probing set S by selecting all paths not subsumed by any previously verified normal path (lines 4 to 8). Once the complete set S for the current round is determined, all paths in S are probed in batches. Paths exhibiting abnormal responses are added to $\mathcal{A}$, while the rest are added to the normal set $\mathcal{N}$ (lines 9 to 12). The iterations continue until no further candidate paths remain. At each iteration, the diagnosis uses only measurements collected within the same time window. By giving preference to longer paths and discarding shorter paths that are covered by them,

the algorithm achieves efficient localization of anomalies and avoids redundant probes.

Failure propagation across slices. Failures within a single slice frequently lead to subsequent effects in other slices. This behavior arises because slices share links, and faults on these shared paths can influence the performance of multiple slices at the same time, for example link A-B in Fig. 8. Therefore, the progressive probing strategy must account for dependencies among slices. The core design follows two rules. First, links or devices that already exhibit failures in other slices are given priority in probing. Second, links or devices that have not yet been examined in other slices are also assigned higher probing priority. These rules ensure that the probing plan remains efficient while detecting failures that may propagate through several slices.

Parallel processing. We advocate a progressive probing strategy executed in batches. Since diagnostics for different slices are mutually independent, probes corresponding to multiple slices within the same batch can be conducted in parallel. Although employing finer-grained progressive probing, such as issuing a single probe at a time, can further minimize probing overhead, it simultaneously diminishes system-level parallelism.

IV. EVALUATION

This section evaluates the diagnostic performance and system efficiency of SRmesh compared to state-of-the-art methods in multiple networks. We evaluated SRmesh considering two complementary sets of experiments. The first set demonstrates the algorithm's diagnostic accuracy and scalability, contrasting our SR-based probing and multipath inference with baseline schemes (Section IV-B). The second set validates the practical workflow and resource overhead of the system prototype in an emulated environment (Section IV-C). Next, we describe the experimental setup used for the evaluation (Section IV-A).

A. Experimental Settings

Scenarios. For performance comparison of large-scale SRv6 networks, we use REPETITA [36], a framework for repeatable traffic engineering algorithm experiments on a dataset of real networks. We use 247 topologies with SR TE that are computed by standard SRLS [37] (default path computation and optimization settings). These selected topologies contain no missing information and each comprises at least ten nodes. In addition, several common topologies are collected as supplementary references; however, they serve solely as structural comparisons and do not include SR TE configurations. We assume that each router connects to a host available for probing. Since our target is a self-managed intra-domain SRv6 network, this requirement is quite practical. Even if a few newly deployed routers lack attached servers, our system supports partial deployment. The overhead for such cases is already analyzed in Section III-A and will not be discussed again here.

Testbed. We deploy SRmesh on a testbed running on a physical server that is equipped with Intel(R) Xeon(R) Gold 6240R processors providing 96 logical cores and 512 GB of memory. The testbed includes 40 container-based routing nodes, and each node runs an instance of FRRouting [38] with

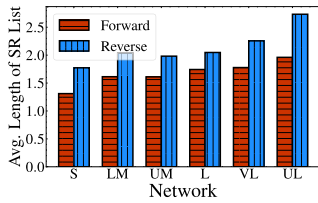


Fig. 9. Length of SR lists.

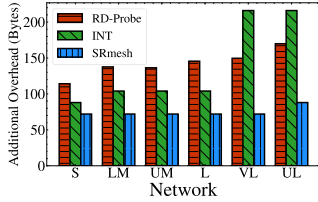


Fig. 10. Header overhead.

native SRv6 forwarding support. The containers are interconnected according to the topology specified by FITI [39]. Every node operates OSPF as its intra-domain routing protocol and is configured to redistribute both local and connected routes. Besides data plane connectivity, all containers also maintain a control plane link to a centralized controller.

Baselines. We compare SRmesh against Pingmesh [18] and selected components of RD-Probe [23]. Other systems do not satisfy the requirements in all experimental scenarios. Their core assumptions differ from those of our setting, which makes direct comparison unsuitable. For example, Zoonet [17] assumes specific inter-region structures that do not extend to arbitrary topologies. Some systems [20], [22] concentrate on loss diagnosis rather than latency bottlenecks. SCMon [19] modifies IGP weights to suppress multipath, which is incompatible with the conditions assumed in our experiments. Pingmesh represents a general state-of-the-art closed-box monitoring system with large-scale deployment capability. In addition, certain parts of RD-Probe can be adapted, after modification, to realize some of the required functions.

B. Performance Evaluation

Probe overhead in SR-based probing. To evaluate the deterministic SR probing mechanism, we first measure the SRH header overhead on production topologies. Each probe follows a forward path configured by SR TE, and the reverse path is constructed by the deterministic SR probe. The SRH on the reverse path contains between 2 and 6 SIDs, and the average SRH overhead is about 80 bytes, as shown in Fig. 9. Even with this additional SRH, all probe packets remain smaller than the 1500-byte MTU, so they are compatible with typical forwarding devices.

For efficiency comparison, we use RD-Probe [23], which builds reverse paths through recursive IP-in-IP encapsulation, and an in-network measurement method based on in-band network telemetry. Fig. 10 shows that RD-Probe introduces on average 176.55 bytes of header overhead for each probe packet, which is nearly twice that of SRmesh, while in-band network telemetry incurs 88–216 bytes depending on the path length. This corresponds to an average header reduction

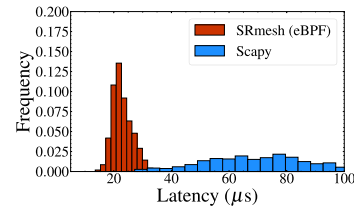
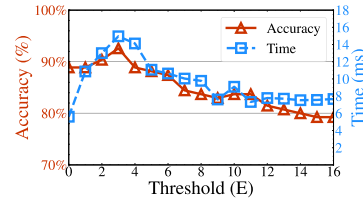


Fig. 11. Operational delay.

Fig. 12. Accuracy and runtime vs. threshold E .

of 52.7% over RD-Probe, and SRmesh also achieves lower header overhead than in-band network telemetry across all evaluated settings. The gain is more evident in ECMP-rich configurations. In this setting, IP-in-IP schemes must encode all reverse paths explicitly, and the overhead of in-band network telemetry grows with the number of traversed hops. The resulting reduction in probe bandwidth lowers the hardware requirement of the sending host by about 50%, depending on the payload size.

High-performance TCP probe via eBPF. We compare the probe latency of a user space Scapy implementation with a kernel space eBPF implementation. The eBPF version reduces latency by 70% and improves the responsiveness of the SRmesh agent. In large SR networks with thousands of slices and tens of probes per node, this latency reduction supports scalability. High-performance UDP and ICMP probing tools [31] are already available, and SRmesh can reuse these tools when necessary.

Sensitivity to hyperparameter E . We evaluate the sensitivity of diagnostic accuracy and runtime to hyperparameter E using the BC network topology. Each link latency is independently sampled from a uniform distribution $\mathcal{U}(10, 100)$ ms and perturbed with Gaussian noise with a standard deviation of 4. Fig. 12 presents the accuracy and execution time as E varies from 0 to 16. As E increases from 0 to 4, diagnostic accuracy improves to a peak of 93.7%, while running time increases to 15.0 ms. For $E > 4$, accuracy gradually decreases to 79.3%, and runtime decreases to 7.7 ms.

These results show that the method is robust to moderate variations in E . A small E makes grouping difficult, while a large E results in loss of accuracy. The turning point $E = 4$ coincides with the standard deviation of the injected noise. Based on this observation, we recommend setting E as the empirical standard deviation of network latency.

Robustness under injected outliers. We examine bottleneck identification when a single outlier is introduced into end-to-end delay measurements. The true configuration contains one bottleneck link with an additional 100 ms of delay. For each outlier level, one path that does not traverse the true bottleneck is selected, and its latency is increased by a specified relative factor. All other paths follow the baseline measurement noise model (noise with a range of $\pm 30\%$).

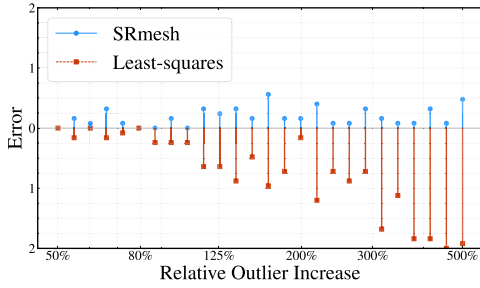


Fig. 13. Error comparison under inserted outliers.

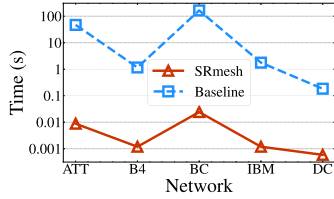


Fig. 14. Greedy vs. ILP scheduling runtime.

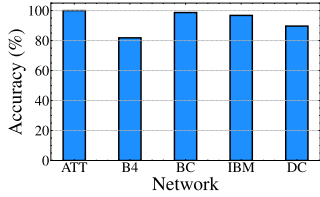


Fig. 15. Accuracy of multipath inference.

SRmesh and a least-squares estimator are applied to the same measurements, and all single-link bottleneck hypotheses are ranked according to their respective objective values.

Figure 13 shows the mean deviation of the true bottleneck's rank from the optimal position (repeat ten times). Across the tested outlier levels, the SRmesh curve remains close to zero, indicating limited change in the inferred ranking. The least-squares curve departs from zero once the outlier reaches moderate intensity and continues to do so as the outlier grows. These results indicate that the robust formulation preserves the correct bottleneck ordering more reliably than least-squares in the presence of isolated measurement outliers.

Optimization efficiency. We evaluate multipath inference of SRmesh against a baseline defined by an integer linear programming-based (ILP-based) optimal solver implemented in Python. Since the baseline does not scale (although SRmesh supports it), we restrict the comparison to five representative topologies with uncertain constraints. To make the baseline solvable within a practical time budget, we further simplify the setting by removing jitter (and $E = 0$) and sampling link latencies from $\mathcal{U}(20, 60)$ ms, thus reducing the complexity. Under these simplified conditions, greedy inference achieves per-slice runtimes of up to 0.024 s, while the baseline requires up to 164.9 s, yielding an average speedup of approximately 3000 $\times$, as shown in Fig. 14. When extrapolated to a network with 4096 slices, the total run times are approximately 29 s for SRmesh versus 49 h for the baseline. Despite this reduction in runtime, the inference fidelity of SRmesh remains high.

TABLE II
DISTRIBUTION OF NETWORK TOPOLOGIES BY SIZE

Category	Node Range	Number of Topologies
Small	10 to 20	51
Lower Medium	20 to 30	64
Upper Medium	30 to 50	58
Large	50 to 70	43
Very Large	70 to 100	15
Ultra Large	100+	16

We treat all SID pairs with multipath ambiguity as the total set and consider a forwarding path inferred correctly as a positive sample. The accuracy is calculated as the ratio of positive samples to the total number of samples. As shown in Fig. 15, SRmesh achieves a diagnostic accuracy between 81.8% and 100%, with a mean of 93.6%. These results demonstrate that SRmesh delivers near-optimal diagnostic quality while remaining computationally efficient and scalable, especially in large-scale scenarios where the baseline becomes computationally infeasible.

Diagnostic effectiveness in diverse topologies. We evaluate end-to-end diagnostic overhead in 247 network topologies, as shown in Table III. These topologies serve as the physical infrastructure for our experiments. Based on each topology, we synthetically generate 4,096 logical network slices, where each slice includes approximately 70% of the physical nodes and is guaranteed to be connected. To model inter-slice fault propagation, we assume that with a probability of 3%, a failure in one slice may affect others. All experiments are repeated 10 times and average results are reported. The baseline for comparison is the full-mesh probing approach exemplified by Pingmesh [18], which can generate up to 50 MB of probe traffic per measurement round.

SRmesh proceeds in three optimization stages. In Step 1, partition-based pruning reduces probe overhead by an average of 59.4%. The effect is especially pronounced in topologies with strong structural modularity: for networks with more than 50 nodes, the reduction reaches up to 80%. In contrast, limited benefit is observed in the data center topology (DC), which lacks a clear partition structure. Building on this, Step 2 applies path-aware pruning, which exploits shared path segments to further reduce probe count by 66.3% on average. This step proves particularly effective in networks with longer end-to-end paths, consistent with the intended design. Finally, Step 3 introduces failure-aware pruning by simulating ten slice-related link failures per topology. This phase yields an additional 51.1% reduction, though the benefit diminishes in large-scale networks due to the growing number of failure-relevant links that must still be probed. In general, the proposed framework reduces diagnostic overhead by an average of 93.4% relative to Pingmesh, while preserving complete visibility into all bottleneck links. Moreover, SRmesh is topology-agnostic, which makes it applicable to SRv6 networks of any topology, such as ISP networks, SD-WANs, and DCNs.

Progressive diagnosis time. In the experiments, progressive probing typically finishes after two rounds. Only a single outlier in the largest topology requires a third round. As a result, the total time remains within a few seconds, so it does not impose a practical limitation. The worst case arises

TABLE III
STEPWISE PROBE OVERHEAD REDUCTION ACROSS TOPOLOGIES

Topology	# Pingmesh [18]	After Step 1		After Step 2		After Step 3		SRmesh Reduction
		Count	Reduction	Count	Reduction	Count	Reduction	
Small	110,840	65,768	37.7%	22,644	65.5%	10,329	54.6%	90.3%
Lower Medium	273,226	126,871	52.6%	42,571	66.4%	21,021	50.9%	92.1%
Upper Medium	690,966	236,607	64.9%	78,760	66.7%	39,830	49.5%	94.1%
Large	1,683,929	448,770	73.1%	149,672	66.6%	75,331	49.7%	95.5%
Very large	3,129,580	725,115	76.7%	242,120	66.4%	121,552	50.6%	96.1%
Ultra Large (100-150)	8,116,688	1,442,959	82.0%	486,178	66.4%	236,771	50.8%	97.0%
Ultra Large (150-200)	15,671,256	2,406,416	84.3%	812,644	66.2%	394,173	51.4%	97.4%
Ultra Large (200+)	51,141,315	6,693,820	86.9%	2,093,551	68.7%	1,257,452	39.9%	97.5%
Average			↓ 59.4%		↓ 66.3%		↓ 51.1%	↓ 93.4%

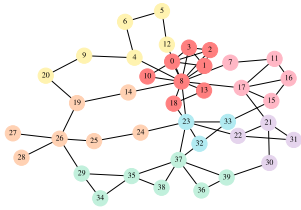


Fig. 16. Partitioning result.

TABLE IV
RELEVANT EXPERIMENTAL DATA IN THE TESTBED

Metric	Value
CPU Utilization	1.2%
Inference Accuracy	83%
Diagnosis Accuracy	100%
Minimum Probe Count	86
Maximum Probe Count	125

when a large number of links fail, but this case is a network-wide problem rather than the bottleneck that SRmesh aims to address.

C. Functional Evaluation

Functional deployment. A prototype of the SRmesh is deployed on our testbed. The agent component is implemented in Go and C, and the server-side code is implemented in Python. Within the FITI topology [39], node partitioning follows the configuration shown in Fig. 16. To validate the diagnosis, latency anomalies are injected at random, each introducing an additional 50 ms of latency along selected links. Under these conditions, all injected failures are diagnosed by SRmesh, and no false positives are observed.

Summary. To evaluate the efficiency of SRmesh under real-world operating conditions, key performance metrics are recorded during multiple anomaly injection experiments. The measurements summarized in Table IV show that SRmesh maintains low resource consumption, with agent-side CPU utilization remaining below 1.2%. This observation indicates the feasibility of deploying SRmesh in resource-constrained environments or as a background service on production routers.

The inference module achieved an average accuracy of 83%, and the diagnosis attains 100% accuracy. When an inference is incorrect, a new round of progressive probing is triggered and produces correct results. The probing cost is on the order

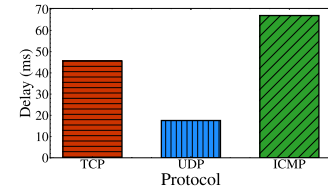


Fig. 17. Results in different protocols.

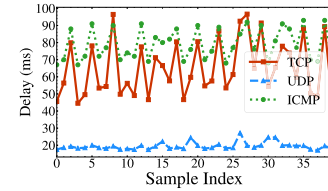


Fig. 18. Delays in different protocols.

of 100. This aligns with the theoretical expectations of the design, where targeted probing avoids exhaustive path enumeration. Overall, SRmesh achieves a balance among diagnostic precision, resource efficiency, and operational scalability.

In the experimental environment, probing different protocols yields different results. As shown in Fig. 17, the average latency of ICMP cannot reflect the latency of either TCP or UDP traffic. Only by using multi-protocol probing in parallel can the full picture of the network be obtained, as shown in Fig. 18.

V. RELATED WORKS

Active network monitoring. Active measurement techniques have been extensively investigated in traditional networks [15], [18], [19], [20], [22], [23], [24] and were later adapted to virtualized environments in order to accommodate multi-tenant architectures and overlay networks [16], [17], [21]. With the increasing adoption of RDMA, several recent efforts design probe approaches tailored to RDMA networks [40], [41], [42], [43]. However, in SRv6 networks the assumptions of these active probes are undermined, which makes existing solutions insufficient for accurate performance diagnosis.

SR performance measurement. Recent works [30], [31], [32], [44], [45] extend active performance measurement protocols to support SR networks. The Simple Two-Way Active Measurement Protocol (STAMP) [46] is enhanced with SR

capabilities in both SR-MPLS and SRv6 data planes [47]. SRv6-PM [48] focuses only on packet loss measurement using alternate marking. Furthermore, SR-STAMP [49] leverages STAMP extensions to measure one-way and two-way latency over SRv6 paths without requiring clock synchronization.

VI. CONCLUSION

In this paper, we propose SRmesh, a distributed probing system for scalable and accurate latency bottleneck diagnosis. First, SRmesh implements distributed probing with eBPF acceleration, ensuring measurement that aligns with actual traffic. Second, SRmesh proposes a latency-based inference algorithm that transforms multipath uncertainty into a solvable constraint system. Third, SRmesh adopts a topology-independent progressive diagnosis strategy to reduce redundancy. Finally, we evaluate it on 247 real network topologies and a prototype implementation is developed. The evaluation results show that SRmesh achieves high diagnostic accuracy while reducing probe overhead by up to 93.4%.

SRmesh currently relies on a server deployed adjacent to each router. For the future, an important extension is to integrate the SRmesh agent into router and switch operating systems, which would broaden the range of deployment scenarios. Another direction is to improve active probing using data-driven methods, including large language models. At present, SRmesh reduces probing overhead through algorithmic optimization. A data-driven probing framework could instead incorporate richer network context to guide the generation of more targeted probe packets.

REFERENCES

- [1] S. Matsushima, C. Filsfils, Z. Ali, Z. Li, K. Rajaraman, *SRv6 Implementation and Deployment Status*. [Online]. Available: <https://datatracker.ietf.org/doc/draft-matsushima-spring-srv6-deployment-status/15/>
- [2] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," RFC Editor, Fremont, CA, USA, Tech. Rep. RFC 8402, 2018.
- [3] (2025). *Connectivity Meets Customization With 5G Network Slicing*. [Online]. Available: <https://cradlepoint.com/resources/blog/connectivity-meets-customization-with-5g-network-slicing/>
- [4] (2025). *How 5G is Reshaping Connected Health Care: A Guide*. [Online]. Available: <https://www.telit.com/blog/5g-connected-health-care-guide/>
- [5] (2025). *Why Low-Latency Communication Matters for Vehicles*. [Online]. Available: <https://horizonconnect.de/low-latency-communication-connected-vehicles/>
- [6] B. Claise, "Cisco systems netflow services export version 9," RFC Editor, Fremont, CA, USA, Tech. Rep. RFC 3954, 2004.
- [7] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *Proc. 11th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Apr. 2014, pp. 71–85.
- [8] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, *Simple Network Management Protocol (SNMP)*, document Rfc1157, 1990.
- [9] Y. Lei, Y. Zhou, Y. Lin, M. Xu, and Y. Wang, "DOVE: Diagnosis-driven SLO violation detection," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–11.
- [10] Y. Zhou et al., "Flow event telemetry on programmable data plane," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 76–89.
- [11] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic in-band network telemetry," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 662–680.
- [12] Q. Huang, H. Sun, P. P. C. Lee, W. Bai, F. Zhu, and Y. Bao, "OmniMon: Re-architecting network telemetry with resource efficiency and full accuracy," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 404–421.
- [13] K. Yang et al., "SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 11, pp. 2876–2894, Nov. 2023.
- [14] H. Zheng et al., "FlyMon: Enabling on-the-fly task reconfiguration for network measurement," in *Proc. ACM SIGCOMM Conf.*, Aug. 2022, pp. 486–502.
- [15] Z. Liu, J. Bi, Y. Zhou, Y. Wang, and Y. Lin, "NetVision: Towards network telemetry as a service," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2018, pp. 247–248.
- [16] C. Fang et al., "Vtrace: Automatic diagnostic system for persistent packet loss in cloud-scale overlay network," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, 2020, pp. 31–43.
- [17] S. Zhu et al., "Zoonet: a proactive telemetry system for large-scale cloud networks," in *Proc. 18th Int. Conf. Emerg. Netw. Exp. Technol.*, 2022, pp. 321–336.
- [18] C. Guo et al., "Pingmesh: A large-scale system for data center network latency measurement and analysis," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 139–152, Aug. 2015.
- [19] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure, "SCMon: Leveraging segment routing to improve network monitoring," in *Proc. INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
- [20] B. Arzani et al., "007: Democratically finding the cause of packet drops," in *Proc. 15th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Apr. 2018, pp. 419–435.
- [21] A. Roy et al., "Cloud datacenter SDN monitoring: Experiences and challenges," in *Proc. Internet Meas. Conf.*, Oct. 2018, pp. 464–470.
- [22] C. Tan et al., "NetBouncer: Active device and link failure localization in data center networks," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 599–614.
- [23] R. Ding et al., "RD-probe: Scalable monitoring with sufficient coverage in complex datacenter networks," in *Proc. ACM SIGCOMM Conf.*, Aug. 2024, pp. 258–273.
- [24] Y. Peng, J. Yang, C. Wu, C. Guo, C. Hu, and Z. Li, "DeTector: A topology-aware monitoring system for data center networks," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2017, pp. 55–68.
- [25] D. Katz and D. Ward, *Bidirectional Forwarding Detection (BFD)*, document RFC 5880, Jun. 2010. [Online]. Available: <https://www.rfc-editor.org/info/rfc5880>
- [26] C. Pignataro, D. Ward, N. Akiya, M. Bhatia, and S. Pallagatti, *Seamless Bidirectional Forwarding Detection (S-BFD)*, document RFC 7880, Jul. 2016. [Online]. Available: <https://www.rfc-editor.org/info/rfc7880>
- [27] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacifico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, "Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–36, Jan. 2020.
- [28] K. Zhao et al., "SRmesh: Deterministic and efficient diagnosis of latency bottleneck links in SRv6 networks," in *Proc. IEEE 33rd Int. Conf. Netw. Protocols (ICNP)*, Oct. 2025, pp. 1–12.
- [29] E. C. Molero, S. Vissicchio, and L. Vanbever, "FAst in-network GraY failure detection for ISPs," in *Proc. ACM SIGCOMM Conf.*, Aug. 2022, pp. 677–692.
- [30] S. Hegde, K. Arora, M. Srivastava, S. Ninan, and N. K. Nainar, *Mechanisms for MPLS Ping and Traceroute Procedures in Inter-Domain Segment Routing Networks*, document RFC 9716, Feb. 2025. [Online]. Available: <https://www.rfc-editor.org/info/rfc9716>
- [31] L. Gong, C. Lin, Y. Qiu, *Ping Path Consistency Over SRv6*. [Online]. Available: <https://datatracker.ietf.org/doc/draft-gong-spring-ping-path-consistency-over-srv6/04/>
- [32] Z. Ali, C. Filsfils, S. Matsushima, D. Voyer, and M. Chen, *Operations, Administration, and Maintenance (OAM) in Segment Routing Over IPv6 (SRv6)*, document RFC 9259, Jun. 2022. [Online]. Available: <https://www.rfc-editor.org/info/rfc9259>
- [33] C. Li, M. Chen, W. Cheng, R. Gandhi, *Path Computation Element Communication Protocol (PCEP) Extensions for Associated Bidirectional Segment Routing (SR) Paths*. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-pce-sr-bidir-path/15/>
- [34] *IP in IP Tunneling*, document RFC 1853, Oct. 1995. [Online]. Available: <https://www.rfc-editor.org/info/rfc1853>
- [35] S. Amante, J. Rajahalme, B. E. Carpenter, and S. Jiang, *IPv6 Flow Label Specification*, document RFC 6437, Nov. 2011. [Online]. Available: <https://www.rfc-editor.org/info/rfc6437>
- [36] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017, *arXiv:1710.08665*.

- [37] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: sub-second optimization for segment routing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
- [38] (2025). *Frouting: Free Range Routing Suite*. [Online]. Available: <https://frouting.org/>
- [39] (2023). *World's First 1.2Tb Ultra-High-Speed Backbone Network Opens in China*. [Online]. Available: <https://www.globaltimes.cn/page/202311/1301765.shtml>
- [40] K. Liu et al., "R-pingmesh: A service-aware RoCE network monitoring and diagnostic system," in *Proc. ACM SIGCOMM Conf.*, 2024, pp. 554–567.
- [41] K. Liu et al., "Hostmesh: Monitor and diagnose networks in rail-optimized RoCE clusters," in *Proc. 8th Asia-Pacific Workshop Netw.*, 2024, pp. 122–128.
- [42] C. Guo et al., "RDMA over commodity Ethernet at scale," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 202–215.
- [43] T. Ma et al., "X-RDMA: Effective RDMA middleware in large-scale production environments," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Nov. 2019, pp. 1–12.
- [44] M. Chen, D. Voyer, R. Gandhi, B. Janssens, and C. Filsfils, "Performance measurement using TWAMP light for segment routing networks," Internet Eng. Task Force, Fremont, CA, USA, Tech. Rep. draft-gandhi-spring-twamp-srpm-10, 1998.
- [45] M. Chen, P. L. Ventre, C. Filsfils, R. Gandhi, and S. Salsano, "Performance measurement using UDP path for segment routing networks," Internet Eng. Task Force, Fremont, CA, USA, Tech. Rep. draft-gandhi-spring-rfc6374-srpm-udp-02, 2019.
- [46] G. Mirsky, G. Jun, H. Nydell, and R. F. Foote, *Simple Two-Way Active Measurement Protocol*, document RFC 8762, Mar. 2020. [Online]. Available: <https://www.rfc-editor.org/info/rfc8762>
- [47] R. Gandhi, C. Filsfils, M. Chen, B. Janssens, and R. F. Foote, *Simple Two-Way Active Measurement Protocol (STAMP) Extensions for Segment Routing Networks*, document RFC 9503, Oct. 2023. [Online]. Available: <https://www.rfc-editor.org/info/rfc9503>
- [48] P. Loreti et al., "SRv6-PM: A cloud-native architecture for performance monitoring of SRv6 networks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 611–626, Mar. 2021.
- [49] C. Scarpitta, G. Sidoretti, A. Mayer, S. Salsano, A. Abdelsalam, and C. Filsfils, "High performance delay monitoring for SRv6-based SD-WANs," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 1, pp. 1067–1081, Feb. 2023.



Yao Tong received the B.Eng. degree from Tsinghua University in 2025, where she is currently pursuing the M.S. degree with the Department of Computer Science and Technology. Her research interests include network measurement, network routing, and network fault awareness.



Yahui Li received the B.S. degree from the School of Software Engineering, Jilin University, Changchun, China, and the Ph.D. degree from Tsinghua University, Beijing, China. She is currently with the Institute for Network Sciences and Cyberspace, Tsinghua University. Her research interests include computer network systems and network security.



Xingang Shi (Member, IEEE) received the B.S. degree from Tsinghua University and the Ph.D. degree from The Chinese University of Hong Kong. He is currently with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network measurement and routing protocols.



Zhiliang Wang (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer science from Tsinghua University, China, in 2001, 2003, and 2006, respectively. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include formal methods and protocol testing, next-generation internet, and network measurement.



Kaiyang Zhao received the B.S. degree from the Department of Computer Science, Shandong University, China, in 2021. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University. His research interests include routing protocol, network measurement, and next-generation internet.



Xia Yin (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer science from Tsinghua University, China, in 1995, 1997, and 2000, respectively. She is currently a Full Professor with the Department of Computer Science and Technology, Tsinghua University. Her research interests include future internet architectures, formal methods, protocol testing, and large-scale internet routing.



Han Zhang (Member, IEEE) received the B.S. degree in computer science and technology from Jilin University, Changchun, China, and the Ph.D. degree from Tsinghua University, Beijing, China. He is currently with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include computer network systems and network security.



Jianping Wu (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees from Tsinghua University. He is currently a Full Professor and the Director of the Network Research Center and a Ph.D. Supervisor with the Department of Computer Science and Technology, Tsinghua University. Since 1994, he has been In-Charge of China Education and Research Network (CERNET). His research interests include next-generation internet, IPv6 deployment and technologies, and internet protocol design and engineering. He is a member of the Information Advisory Committee, the Office of the National Information Infrastructure, a Secretariat of the State Council of China, and a Vice President of the Internet Society of China (ISC).