

SRmesh: Deterministic and Efficient Diagnosis of Latency Bottleneck Links in SRv6 Networks

Kaiyang Zhao^{*}, Han Zhang^{†‡}, Yao Tong^{*}, Yahui Li[†], Xingang Shi^{†‡}, Zhiliang Wang^{†‡}, Xia Yin^{*‡}, Jianping Wu^{*‡‡}

^{*}Department of Computer Science and Technology, Tsinghua University, Beijing, China

[†]Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China

[‡]Zhongguancun Laboratory, Beijing, China

Abstract—Segment Routing over IPv6 (SRv6) has attracted more attention from network operators. Diagnosing performance bottlenecks for SRv6 tunnels is critical to maintaining network quality. However, SRv6 introduces priority policies, special forms of multipath routing, and SR-based network slicing, all of which make existing methods difficult to apply. In this paper, we present **SRmesh**, a framework for diagnosing latency bottleneck links specifically tailored for SRv6 tunnel performance analysis. First, we adopt SR-based probing to deterministically control routing paths and ensure that probe packets emulate real production traffic. Second, we employ a latency-based multipath inference to resolve routing ambiguities caused by SR. Third, we introduce a topology-independent 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 real network topologies. The results indicate that **SRmesh** achieves high diagnostic accuracy with up to a 91.9% reduction in probe overhead, demonstrating its practicality and scalability in large-scale SRv6 environments.

I. INTRODUCTION

Segment Routing (SR), as a source routing protocol, has been widely adopted in various networks [1], such as Internet Service Provider (ISP) networks and Software-Defined Wide Area Networks (SD-WANs). These networks require fine-grained traffic engineering (TE) [2] and use SR over IPv6 (SRv6) to deliver multiple network slices, each configured to meet specific Service Level Objectives (SLOs), including latency demands. For instance, latency-sensitive slices are provisioned for services such as remote healthcare [3], e-finance [4], and vehicular communications [5]. Operators must identify and rectify bottlenecks if the end-to-end latency exceeds established thresholds.

Diagnosing latency bottlenecks in SRv6 networks remains challenging due to several issues. First, it is difficult to precisely reflect the behavior of actual service traffic. Measurement methods must prevent the generation of misleading results, which may arise from factors such as packet prioritization and firewall filtering. The above phenomena are especially common in SRv6 networks supporting multiple slices. Second, multipath routing from SR complicates the diagnostic process. The algorithm must be capable of precisely locating the bottleneck link among multiple potential paths.

Third, the solution must maintain low overhead to remain practical in large-scale SRv6 overlay networks.

Existing SR path quality monitoring approaches fall into two categories: passive and active measurement. However, passive approaches [6]–[14] are unable to provide early warnings for links that may experience future latency anomalies (e.g., backup or rerouted paths during failures), nor can they offer reliable performance estimates required for dynamic SR TE optimization. On the other hand, active approaches can be classified into in-network and host-based approaches, based on the necessity for modifications to network devices. In-network approaches [15]–[17] require alterations to all forwarding devices, rendering them impractical for SR deployments that seek to facilitate incremental implementation. One of the motivations for adopting SR is to avoid modifications to existing infrastructure. Meanwhile, host-based approaches [18]–[24] often incur substantial probing overhead, and the probe packets may not traverse the same forwarding paths as actual business traffic. Furthermore, multipath routing under SR introduces additional complexity in bottleneck diagnosis.

Vision. We aim to develop a lightweight, accurate, and fast tool for locating link-level latency bottlenecks within SR tunnels in SRv6 networks. Similar to BFD [25], [26], a primary application of such a tool would be to enable rerouting decisions once a bottleneck is identified. In addition, the latency performance measurements obtained during the probing process can serve as precise inputs for rerouting orchestration, improving the accuracy of TE decisions.

Problem Statement. We focus on the following problem:

Can we design a system tailored to SRv6 that identifies latency bottlenecks within SR tunnels?

A latency bottleneck link is defined as a link whose performance violates predefined latency objectives. In a well-functioning network, the latency of each link tends to remain stable over time. These latency objectives can be derived from empirical measurements (multiplying periodic latency values with a redundancy factor) or be statically configured by network operators. Notably, the scope of this work does not extend to root cause analysis or remediation mechanisms.

SRmesh. We propose **SRmesh**, a host-based system specifically designed for SRv6 networks to diagnose link-level latency bottlenecks. In contrast to existing approaches,

SRmesh achieves precise identification of problematic links while maintaining low diagnostic overhead. To achieve these goals, SRmesh facilitates both efficient probing and accurate diagnosis with the following **novel designs**:

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

Second, we propose the **multipath inference** to estimate Equal Cost Multipath (ECMP) routing selections in SR by minimizing discrepancies from end-to-end latency measurements (Section III-B). The design transforms routing uncertainty into a constraint system and employs a greedy algorithm to group candidate latencies, facilitating the inference of actual path selection.

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

In summary, our **contributions** are summarized as follows:

- We conduct a wide study of the new requirements imposed by latency bottleneck links diagnosis in the SRv6 network (Section II).
- We design and implement SRmesh, a scalable and accurate diagnosis framework for SRv6 networks. It integrates deterministic 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 91.9%. To further validate the practical viability of SRmesh, we deploy a prototype in a controlled testbed environment (Section IV).

We open-source our SRmesh prototype at <https://github.com/DeepShield-AI/SRmesh>.

II. BACKGROUND AND MOTIVATION

A. Operator Concerns in SRv6 Networks

SR is a source routing mechanism that encodes path nodes or network functions as segments encapsulated in the Segment Routing Header (SRH) of the packet. In SRv6, a Segment Identifier (SID) is a 128-bit address that may include a slice ID, allowing the SID to indicate both forwarding behavior and slice semantics. The ingress node inserts the SRH, and packets are forwarded along the segment list, with each intermediate node interpreting the active segment as the destination address.

The operational simplicity of SRv6 renders it effective for dynamic network environments such as SD-WAN [1]. However, these deployments are susceptible to gray failures,

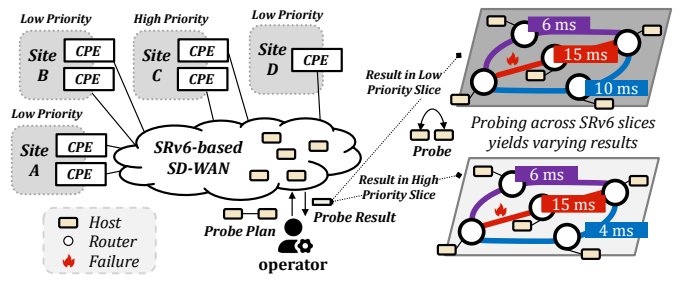


Fig. 1: Case Study: Diagnosing in SRv6-based SD-WAN. Sites connect through Customer Premises Equipment (CPE) devices, which assign traffic to slices with different priorities. Probing only low-priority slices yields erroneous results.

which degrade performance by introducing abnormal latencies without causing full service interruptions [28]. One of the advantages of SRv6 is its ability to quickly switch paths in the event of a failure [29], but this can also result in latency fluctuations, particularly during the failure itself. Moreover, incorrect SR policies and failure to properly configure Quality of Service (QoS) policies can negatively impact latency performance.

B. New Requirements for Diagnosis in SRv6 Networks

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

In addition, SRv6-based slicing exacerbates this issue. As shown in Fig. 1, a router failure may trigger rerouting based on probing traffic from a low-priority slice, which reports a 6 ms path. However, the optimal 4 ms path for the high-priority slice is disregarded. Such divergence leads to suboptimal decisions and degraded service. To prevent suboptimal routing decisions, SR measurements must correspond with actual traffic behavior.

Challenge for RI: Determinism requires the completion of SR-level probing and the carrying of the corresponding SRH. Moreover, probing necessitates the control of both forward and reverse paths utilizing SRHs. This progress introduces additional packet headers, increasing the risk of exceeding the Maximum Transmission Unit (MTU).

Requirement 2: Fast and accurate diagnosis of bottleneck links in multipath routing. In SRv6 networks, multipath routing introduces uncertainty in the routing paths of probe packets. The exact path of each probe is unknown, complicating the correlation of observed performance metrics with particular links. SR-based multipath is different from

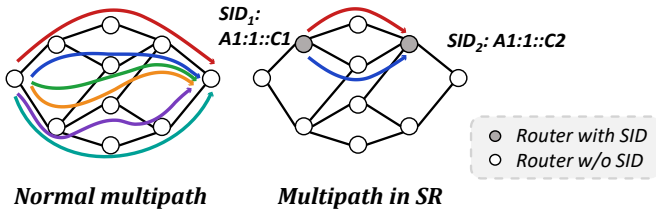


Fig. 2: Multipath diversity in normal IP routing vs. SR. SID-related constraints limit path options in SR.

normal multipath, as shown in Fig. 2. In SR, the source node specifies a set of intermediate nodes that the packet must pass through. This mechanism allows the source to control the routing path. However, between any two segments in the path, multiple paths still exist. Such multipath behavior only occurs between nodes that have SIDs. This characteristic provides additional structural information that can be used to address the multipath diagnosis problem.

Moreover, SR paths are unidirectional, and when bidirectional SR lists are constructed [33], multipath between SIDs can introduce asymmetry. It undermines the assumption that one-way latency is equivalent to half the Round-Trip Time (RTT), leading to non-deterministic path measurement.

Challenge for R2: SR introduces path variability as a result of ECMP among SIDs. Consequently, diagnosis must account for all potential path combinations, which leads to an NP-hard problem as the number of viable paths grows exponentially with network size. Therefore, it is essential to develop efficient polynomial-time approximation algorithms.

Requirement 3: Cost-effective probing of large-scale SRv6 networks. Traditional networks assign each router a distinct IP address. In contrast, SR-based overlay networks introduce a two-dimensional addressing scheme. This scheme uses the address of the physical node combined with a slice ID to identify logical nodes within the slices. Large-scale SRv6 networks, potentially comprising hundreds of physical nodes, may assign a single router to thousands of slices.

Consider an example of probing overhead: In an overlay network with 100 physical routers and 4,096 slices, Pingmesh [18] generates approximately 5,000 probing paths for the physical network but over 20 million probing paths for the overlay network, a $O(10^3)$ increase.

However, a large proportion of these probes are redundant. As illustrated in Fig. 3, under a full-mesh probing scheme, many links are repeatedly measured (even over 1,400). Such redundancy becomes increasingly severe in large-scale topologies and offers little diagnostic value while consuming significant resources.

Challenge for R3: In structured networks such as data center networks (DCNs), the topology exhibits regular patterns that allow efficient probing strategies. In contrast, SRv6 networks have no fixed structural regularity. The challenge is to design scalable algorithms that use a limited number of probes to cover all slices and their SR paths while still ensuring accurate localization of performance bottlenecks.

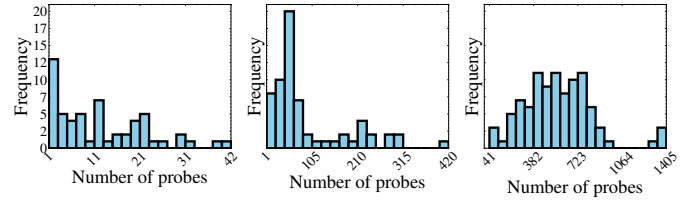


Fig. 3: Probing redundancy under full-mesh SR measurements in AT&T, B4, and VtWavenet topologies. The paths between node pairs follow shortest-path routing, and the redundancy is derived from graph-theoretic analysis.

TABLE I: A summary of the features 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.

C. Motivating SRmesh

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

In-network probing introduces significant overhead and lacks scalability in SRv6 networks. NetVision [15] inserts two label stacks (SR stack and INT stack), which brings additional overhead. During the probe transmission, SR labels are popped to forward probes, while INT labels are pushed to record telemetry data. V-Trace [16] installs rules on virtual forwarding devices to trace packets with fine granularity. This approach requires configuration changes on all forwarding devices, and long-term deployment can reduce forwarding performance, with a 6% drop per rule on average. Zoonet [17] supports hop-by-hop tracing and probe pruning but similarly requires modifying every device in the forwarding path, making it difficult to adopt in SRv6 networks used by ISPs.

Host-based probing lacks determinism and struggles with multipath ambiguity. Pingmesh [18] and VNET-pingmesh [21] perform end-to-end full-mesh probing, but they incur 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 RD-Probe is primarily designed for DCNs. Similar 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, which facilitate the design of probing schemes. SRv6 networks, however, use topologies

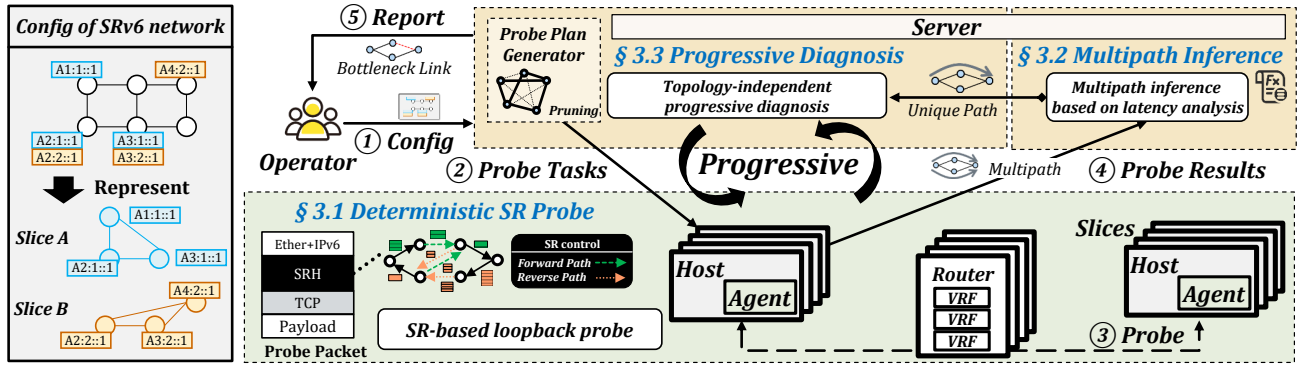


Fig. 4: Overview of SRmesh.

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, since the monitoring plane cannot differentiate among 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.

Relation of SRmesh to network tomography. Network tomography [35], [36] infers internal network states such as latency and packet loss by collecting end-to-end measurements at the network edge. This work is not confined to network tomography, and notable distinctions arise when it is applied to SR networks. First, network tomography does not consider the design of packet generators (Requirement 1). Second, multipath modeling under SR has specific characteristics (Requirement 2). Third, the multi-slice network constructed by SR is not equivalent to performing tomography over multiple networks, where bottlenecks across slices are correlated and not fully independent (Requirement 3).

III. SRMESH FRAMEWORK

We present the overall architecture of the SRmesh in Fig. 4. SRmesh consists of two key components: **agent** and **server**. The **server** receives network configuration inputs from the network operator through an interface. These inputs include the physical topology and the SR configuration, which specifies slice-related information (1).

The SR configuration establishes a series of overlay networks, each corresponding to a specific slice. These overlays create logical paths over 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), even though they traverse the same physical nodes.

Based on the configured slice-specific SIDs, the controller derives a set of slice-aware forwarding paths and distributes corresponding probing plans to the **agents** (2). SRmesh ensures that probes follow the same SID lists as actual service traffic, preserving slice-level path semantics during performance measurement.

The **agent** is deployed on selected container nodes, virtual machines, or physical hosts. It utilizes predefined eBPF hooks and probing extensions to capture probing data (3), which is subsequently transmitted to the **server** (4). This process (2–4) is progressive until the diagnosis is completed. The system outputs the identified bottleneck links, each including its logical representation in the SRv6 slice or policy and the corresponding physical path in the underlying topology (5).

A. Deterministic SR Probe

Accurate bottleneck link diagnosis in SRv6 networks requires deterministic probing. Determinism entails elimination of uncertainty in forward and reverse paths to avoid asymmetry-induced bias and matching the forwarding treatment of production traffic, without interference from priority-based scheduling. Traditional tools such as ping fail to meet these requirements, as they rely on probe packets that are either subject to filtering or are assigned lower forwarding priorities and thus do not accurately represent the behavior of in-tunnel business traffic, especially TCP-based services.

★ **DESIGN 1: SR-based loopback probe.** In response, SRmesh adopts a TCP-based active probing method for SR tunnel quality measurement, which needs to overcome several core challenges. One is path asymmetry, which can lead to measurement errors. The other is scalability, as the baseline method needs to establish thousands of TCP sessions per second, risking exhaustion of resources.

Probe packet. In SRv6 networks, routing between the source and destination SIDs is typically subject to ECMP hash functions based on the five-tuple, often resulting in asymmetrical forward and reverse paths. To address this, we propose a deterministic probing mechanism: at the sending node, the SR list is generated according to the SR configuration and inserted into the packet payload along with a timestamp and the reverse SR list. Upon arrival at the receiving node, before the packet enters the SRv6 processing pipeline, the agent extracts the return SR list from the payload and constructs a new SRH, thereby explicitly specifying the reverse source route. The sender then computes the RTT based on the timestamps. If the ingress router is configured with a SR policy, the host inherits and embeds the corresponding SR list; otherwise, the host

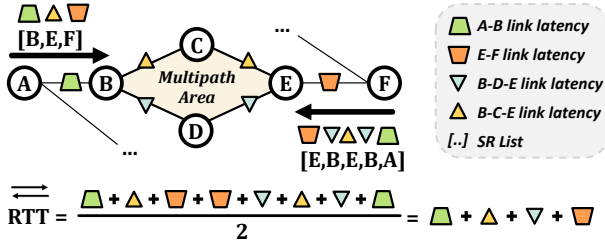


Fig. 5: **SR-based loopback probe.** Reverse SR lists are embedded to enable reverse path traversal, yielding RTT under multipath.

constructs a minimal SR list based on the network topology and configuration.

SRH control. This SR list includes entries for possible reverse paths for the multiple SID pairs. For instance, if multiple paths exist between nodes B and E, a reverse SR list from B to E is inserted between E and B, as in Fig. 5. On the receiver side, the SR list from the payload is incorporated into the packet. Ultimately, the packet will traverse the multiple SID paths. The resulting latency represents the cumulative latency across a sequence of predetermined paths. By excluding the reverse SR list from the forward probe header, the responder agent is enabled to insert it upon return, thereby reducing the overall length of the SR list by half.

TCP acceleration. The agent uses eBPF to hook into kernel-level TCP state transitions, addressing the inefficiencies of traditional TCP probing, as shown in Fig. 6. By intercepting TCP SYN packets before they enter the protocol stack, SRmesh generates and returns RST responses in situ. This design accelerates RTT measurement without the need for complete TCP connections, thereby reducing the overhead associated with traditional TCP connection establishments.

Compared to user-space implementations, eBPF enables high-performance packet processing with lower latency and reduced CPU utilization. In networks comprising up to thousands of slices, acceleration is essential for maintaining responsiveness. Since each TCP connection consumes a file descriptor, maintaining thousands of concurrent connections in SRmesh requires raising descriptor limits to avoid failures due to exhaustion. eBPF-based connectionless TCP probing eliminates this constraint.

B. Multipath Inference

Multipath introduces uncertainty into the end-to-end latency attribution process, as each observed latency measurement may correspond to one of several possible internal paths. The constraint system that models such latency observations thus comprises both deterministic constraints (e.g., algebraic equalities) and uncertain constraints (e.g., disjunctive expressions). Solving this system is computationally intractable, as the number of path combinations increases exponentially with network scale, rendering the problem NP-hard. The model is defined as follows.

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

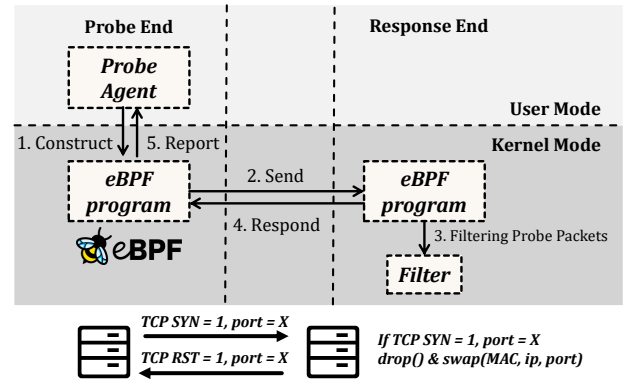


Fig. 6: **TCP Acceleration.** eBPF bypasses TCP kernel handling to accelerate RTT measurement.

Subject to:

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

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

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

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

Existing methods for addressing multipath mostly rely on extensive probing. For instance, different values in the Flow Label [37] field of the IPv6 header are used to trigger different ECMP paths. Then, based on the coupon collector theory, all links are covered. For instance, it may send thousands of packets with different ECMP hash values and assume that such volume achieves complete path coverage. However, such cost-insensitive probing increases the number of probe packets.

★ **DESIGN 2: Multipath inference based on latency analysis.** SRmesh uses an iterative greedy algorithm to facilitate diagnosis in multipath routing scenarios, as outlined in Algorithm 1. The algorithm determines routing paths through the analysis of end-to-end latency measurements, promoting compliance with constraints.

The inference first addresses deterministic constraints $\mathcal{C}_d$ via algebraic elimination, directly obtaining latency values from exact constraints (lines 2–4). Each uncertain constraint c_u consists of pairs (α, κ) , where α is the known latency, and κ is the unknown latency to be estimated. For each constraint c , the set $\mathcal{V}_c$ collects these unknown values for further processing (line 7). Following deterministic resolution, the algorithm calculates candidate latency values for each

Algorithm 1 Greedy inference

Input: $\mathcal{C}_d, \mathcal{C}_u, E$
Output: $\mathcal{S}$: Estimated values to unknown latencies

```

1:  $\mathcal{S} \leftarrow \emptyset$ 
2: for each  $c \in \mathcal{C}_d$  do
3:   Solve  $c$  by elimination, update  $\mathcal{S}$ 
4: end for
5: while  $\mathcal{S}$  updated do
6:   for each  $c_u \in \mathcal{C}_u$  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  $c_u \in \mathcal{C}_u$  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  $\mathcal{S}$ 
  
```

unresolved disjunctive constraint $\mathcal{C}_u$ and organizes them into clusters $\mathcal{G}$ according to a proximity threshold E (line 8). In 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 14-16). This process is iterative; newly resolved values can simplify additional constraints, facilitating further updates. In cases where disjunctive constraints are unresolved post-greedy inference, the algorithm calculates the mean of candidate values for each group and assigns this mean as an approximation (lines 20-22). The process persists until no additional updates can be implemented.

To clarify the detailed procedure, consider the example depicted in Fig. 7. In the presence of disjunctive constraints such as $d_1 + (d_a \vee d_b \vee d_c) = b$, the algorithm evaluates the differences between candidate latency values, such as $|d_a - d_b|$ and $|d_b - d_c|$. If the absolute difference between any two candidates exceeds a specified threshold E , it indicates that the values are associated with distinct routing paths. If the difference is within the threshold, the corresponding values (e.g., d_b and d_c) are grouped, and a representative value is selected to approximate them. The algorithm proceeds in a greedy manner to assign values that reduce the overall approximation error.

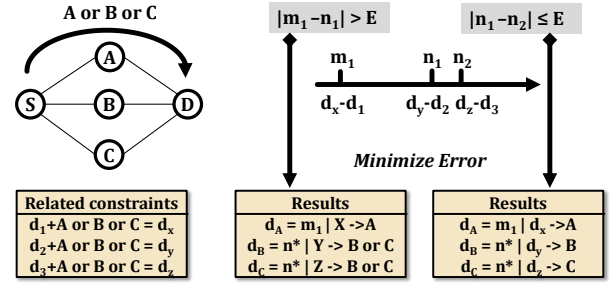


Fig. 7: Example of multipath inference using threshold-based grouping and greedy resolution of disjunctive constraints.

C. Progressive Diagnosis

SRmesh is designed to ensure full path coverage within all topologies while reducing the number of probes required. A naive baseline adopts full-mesh probing for each slice, at the cost of considerable resource consumption. In contrast, the progressive diagnosis applies a partitioned and progressive strategy that preserves coverage with reduced probing overhead. One of the key challenges lies in sustaining this efficiency across varied topologies, where structural differences can affect the effectiveness of probe planning.

★ **DESIGN 3: Topology-independent progressive diagnosis.** First, SRmesh adopts *divide-and-conquer* by partitioning the network into sub-networks with dense intra-partition connections. The algorithm is independent of topology and does not need to rely on a special topological structure. Second, *progressive probing* is employed, where once a long path is validated, shorter paths within the same partition can be inferred, reducing redundant probes. In addition, *cross-slice links* are prioritized for early probing to facilitate the prompt identification of failures that may impact multiple slices, as shown in Fig. 8.

To formalize the problem, a model is introduced for allocating probes across multiple network slices, with a focus on coverage. The objective is to reduce probing overhead while ensuring that inter-slice dependencies are captured and all relevant paths are exercised. The model is defined as follows.

$$\min \sum_{i=1}^N \sum_{u,v \in V_{s_i}} X_{s_i}(u,v) \quad (5)$$

$$\text{Subject to: } Y_{s_i}(u,v) \geq 1, \quad \forall (u,v) \in E_{s_i} \quad (6)$$

$$Y_{s_i}(u,v) = \sum_{u,v \in V_{s_i}} X_{s_i}(u,v) \cdot A_{s_i}(u,v) \quad (7)$$

$$X_{s_i}(u,v) \leq A_{s_i}(u,v), \quad \forall u,v \in V_{s_i} \quad (8)$$

$$X_{s_i}(u,v) + X_{s_i}(v,u) \leq 1, \quad \forall u,v \in V_{s_i} \quad (9)$$

$$X_{s_i}(u,v) \in \{0,1\}, \quad \forall u,v \in V_{s_i} \quad (10)$$

First, objective function (5) 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,

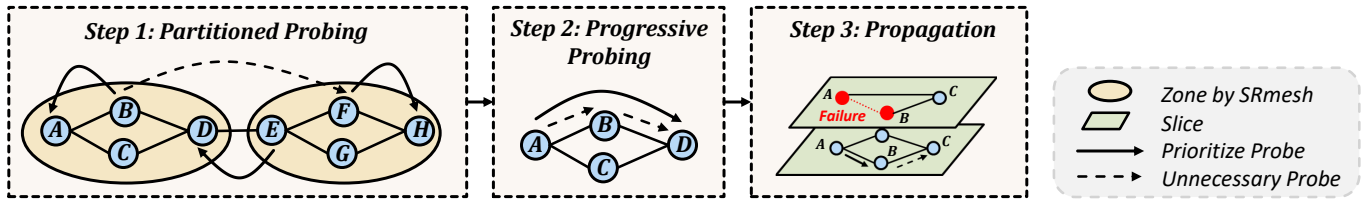


Fig. 8: Workflow of topology-independent progressive diagnosis.

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 Z
- 6: $Z^* \leftarrow \arg \max_Z \Delta Q(v, Z)$
- 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** $\mathcal{Z}$

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** $\mathcal{A}$

full coverage (6) requires that each edge in slice s_i be covered by at least one probing pair. Building on this, definition (7) 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 (8) ensures that probing pairs are only established between nodes with an available routing path. In terms of efficiency, probing pair symmetry (9) restricts the selection to at most one direction for each node pair, thereby preventing redundancy. Finally, binary constraint (10) ensures that a probing pair is either selected or not without intermediate values.

Partitioned probing. Each slice is partitioned into sub-networks with denser intra-partition connections, whereas inter-partition links are highly reused during end-to-end full-mesh probing. This partitioning enables more efficient probing by concentrating on the local connections first, thus reducing the overall number of probes required for complete coverage.

Algorithm 2 formalizes the modularity maximization partitioning approach. Initially, the algorithm sets each node as an independent partition (line 1). It then iteratively optimizes the modularity metric Q defined by Equation (11) through a greedy node-moving strategy. At each iteration, the modularity increment ΔQ is calculated for every node with respect to neighboring partitions. A node v is subsequently moved into the partition P^* that provides the maximal positive modularity increase (lines 7–9). The algorithm continues this iterative refinement until no further positive modularity gain can be achieved, thus indicating convergence (line 3). Upon conver-

gence, intra-partition connectivity is ensured by conducting full-mesh probes within each partition. Inter-partition connectivity is verified by selecting representative nodes (e.g., nodes D and E in Fig. 8) from each partition to participate in a second-stage full-mesh probing.

The modularity Q guiding the partition optimization is formally defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \quad (11)$$

where A_{ij} denotes the adjacency matrix, k_i and k_j represent 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 relies on the principle that a lengthy probe path displaying normal behavior indicates that all shorter paths contained within that probe path exhibit normal behavior. 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). The total number of probes is significantly decreased while preserving accuracy.

Algorithm 3 formalizes the progressive probing strategy. The algorithm begins by initializing 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. In each iteration, it constructs a

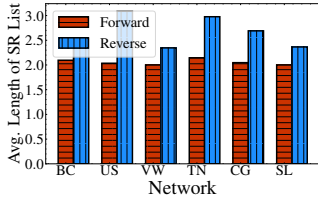
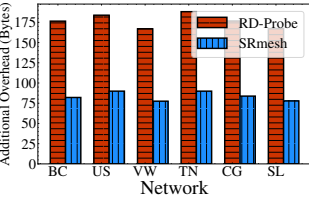
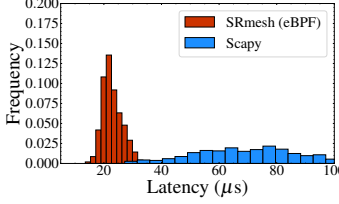
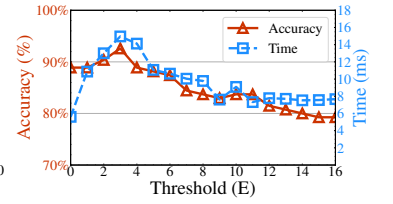


Fig. 9: Length of SR Lists.

Fig. 10: Header overhead.

Fig. 11: Operational delay.

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

candidate probing set S by selecting all paths not subsumed by any previously verified normal path (lines 4–8). Once the full set S for the current round is determined, all paths in S are probed in batch. Paths exhibiting abnormal responses are added to $\mathcal{A}$, while the rest are added to the normal set $\mathcal{N}$ (lines 9–12). This iterative process persists until no additional candidate paths are available. The algorithm prioritizes longer paths and eliminates shorter, covered ones, thereby ensuring efficient anomaly localization and reducing redundant probes.

Failure propagation across slices. Failures in one slice tend to result in cascading impacts on other slices. This issue occurs because of the links among slices, where failures in these shared paths can concurrently affect the performance in multiple slices (e.g., A-B in Fig. 8). Therefore, the progressive probing strategy needs to account for cross-slice dependencies. The core of the design rests on two key rules: (1) links or devices exhibiting failures in other slices should be prioritized for probing, and (2) links or devices that have not yet been probed in other slices should be given higher priority for probing. These rules ensure that the probing strategy is efficient in detecting potential failures that could propagate across multiple slices.

IV. EVALUATION

This section evaluates the diagnostic performance and system efficiency of **SRmesh** against state-of-the-art methods across different networks. In addition, we compare our SR-based probing and multipath inference algorithm against baseline approaches to highlight their respective advantages. Finally, we conduct a functional evaluation on a testbed.

A. Experimental Settings

Scenarios. For performance comparison of large-scale SRv6 networks, we use REPETITA [38], a framework for repeatable traffic engineering algorithm experiments on a dataset of real networks. We use six topologies with SR TE that are computed by standard SRLS [39] (default path computation and optimization settings). In addition, we have collected some common topologies, but these topologies only provide additional topology references and do not include SR TE. Table II lists the number of nodes and the average degree for each selected network.

Testbed. We evaluate **SRmesh** on a testbed deployed on a physical server equipped with Intel(R) Xeon(R) Gold 6240R CPUs (96 logical cores) and 512 GB of RAM. The testbed hosts 40 containerized routing nodes, each running a dedicated

TABLE II: Statistical information of the networks.

Network	Label	Nodes	Average degree	SR TE
AT&T	ATT	25	4.48	None
Google B4	B4	12	3.17	None
IBM	IBM	17	2.70	None
Bell Canada	BC	48	2.67	✓
SpineLeaf	DC	10	3.20	None
VTL Wavenet	VW	92	2.09	✓
US Signal	US	61	2.56	✓
TATA	TN	145	2.57	✓
Cogent	CG	197	2.47	✓
Sprintlink	SL	315	6.17	✓

FRRouting [40] with native support for SRv6 forwarding. These containers are organized according to the topology defined by the FITI [41]. Each node uses OSPF as its intra-domain routing protocol, with configurations that redistribute both local and connected routes. In addition to data-plane connectivity, all containers maintain a control-plane link to a centralized controller.

Baselines. We compare **SRmesh** with Pingmesh [18] and selected components of RD-Probe [23]. Other systems could not meet the requirements in all experimental scenarios. Their core assumptions differ from ours, making direct comparison inappropriate. For instance, Zoonet [17] assumes specific inter-region structures that do not generalize to arbitrary topologies. Some systems [20], [22] focus on loss diagnosis rather than latency bottlenecks. SCMon [19] alters IGP weights to disable multipath, which conflicts with the conditions assumed in our experiments. Pingmesh is a general state-of-the-art black-box monitoring system with large-scale deployment capability. Meanwhile, parts of the RD-Probe can be adapted to achieve some functions after modification.

B. Performance Evaluation

Probe overhead in SR-based probing. In evaluating the deterministic SR probing mechanism, we begin by measuring the SRH header overhead across six production topologies (BC, US, VW, TN, CG, SL). These topologies are selected because they support feasible SR TE configurations. Each probe follows an SR TE-configured forward path, while the reverse path is constructed using our proposed deterministic SR probe. The reverse-path SRH includes between 2 and 6 SIDs and the average SRH overhead is around 80 bytes, as shown in Fig. 9. Despite the added SRH, all probe packets remain within the 1500-byte MTU, ensuring compatibility with typical forwarding devices.

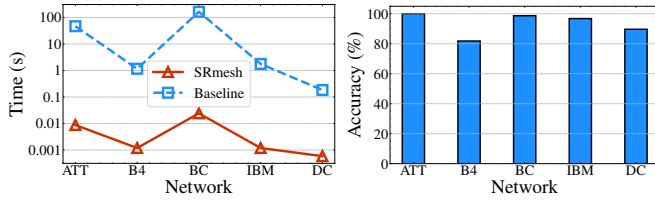


Fig. 13: Greedy vs. ILP Fig. 14: Accuracy of multipath inference.

To quantify efficiency, we compare against RD-Probe [23], which constructs reverse paths using recursive IP-in-IP encapsulation. Fig. 10 shows that, on average, RD-Probe incurs 176.55 bytes of header overhead per probe packet, nearly double that of SRmesh. This yields an average header reduction of 52.7% per packet. The efficiency gain is especially pronounced in ECMP-rich scenarios, where IP-in-IP schemes must encode all reverse paths explicitly. This reduction in probe bandwidth reduces the hardware requirements of the host used to send packets by approximately 50% (depending on the payload size).

High-performance TCP probe via eBPF. We compare the probe latency of a user-space Scapy script and a kernel-space eBPF implementation. The eBPF version reduces latency by 70%, significantly improving the responsiveness of the SRmesh agent. In large-scale SR networks with thousands of slices and tens of probes per node, such latency reduction facilitates scalability. High-performance UDP and ICMP probing tools [31] already exist, and SRmesh can directly leverage these implementations if needed.

Sensitivity to hyperparameter E . We evaluate the sensitivity of diagnostic accuracy and runtime to the 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 runtime rises to 15.0 ms. For $E > 4$, accuracy gradually declines to 79.3%, and runtime decreases toward 7.7 ms.

The findings indicate that the method is robust to moderate variations in E . A small E makes grouping difficult, while a large E results in a 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 to the empirical standard deviation of network latency.

Optimization efficiency. We evaluate multipath inference of SRmesh against a baseline defined by an integer linear programming-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, the 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. 13. When extrapolated to a network with 4096 slices, total runtimes are approximately 29 s for SRmesh versus 49 hours for the baseline. Despite this reduction in runtime, inference fidelity of SRmesh remains high.

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 computed as the ratio of positive samples to the total number of samples. As shown in Fig. 14, SRmesh achieves 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 across diverse topologies. We evaluate end-to-end diagnostic overhead on ten 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 50.6%. The effect is especially pronounced in topologies with strong structural modularity: for networks with more than 50 nodes, the reduction reaches up to 66.6%. 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 66.2% 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 51.4% reduction, though the benefit diminishes in large-scale networks due to the growing number of failure-relevant links that must still be probed. Overall, the proposed framework reduces diagnostic overhead by an average of 91.9% relative to the Pingmesh, while preserving complete visibility into all bottleneck links. Moreover, SRmesh is topology-agnostic, making it applicable to SRv6 networks of any topology, such as ISP networks, SD-WANs, and DCNs.

Progressive diagnosis time. In experiments, progressive probing typically completes within two rounds. Only a single outlier case in the SL (the largest topology) triggered a third round. Consequently, the total time remains on the order of a few seconds, which is not a practical limitation. The worst-case scenario occurs when a large number of links fail, but

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	
ATT	298,143	144,590	51.5%	48,228	66.6%	24,173	49.9%	91.9%
B4	63,842	46,145	27.7%	16,041	65.2%	7,104	55.7%	88.9%
IBM	125,397	70,243	44.0%	24,171	65.6%	11,034	54.3%	91.2%
BC	1,143,631	315,812	72.4%	106,030	66.4%	52,434	50.5%	95.4%
DC	46,530	42,190	9.3%	14,186	66.4%	6,962	50.9%	85.0%
VW	4,309,746	852,225	80.2%	297,553	65.1%	129,728	56.4%	97.0%
US	1,862,768	446,322	76.0%	150,929	66.2%	73,064	51.6%	96.1%
TN	10,755,860	1,770,240	83.5%	610,167	65.5%	277,663	54.5%	97.4%
CG	19,865,998	3,029,980	84.7%	1,049,494	65.4%	470,442	55.2%	97.6%
SL	51,134,486	6,672,313	87.0%	2,087,022	68.7%	1,253,233	40.0%	97.5%
Average			↓ 50.6%		↓ 66.2%		↓ 51.4%	↓ 91.9%

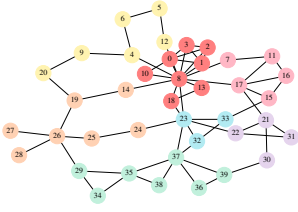


Fig. 15: Partitioning result.

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

this scenario is a network-wide problem, not the bottleneck that SRmesh aims to address.

C. Functional Evaluation

Functional deployment. We deployed a prototype of the SRmesh on our testbed. The agent component is implemented in Go and C, while the server-side code is developed in Python. Within the FITI topology [41], node partitioning is applied as illustrated in Fig. 15. To validate the diagnosis, we randomly injected anomalies of latency, each introducing an additional 50 ms of latency along selected links. All injected failures were diagnosed by the SRmesh, with no false positives observed.

Summary. To assess the efficiency and reliability of SRmesh under real-world operating conditions, we collected key performance metrics during multiple anomaly injection experiments. As summarized in Table IV, SRmesh demonstrates low resource consumption, with agent-side CPU utilization remaining under 1.2%. This result indicates the feasibility of deploying SRmesh in resource-constrained environments or as a background service in production routers.

The inference module achieved an average accuracy of 83%, and the diagnosis achieved 100% accuracy in failure localization. Incorrect inferences trigger a new round of progressive probing, which produces correct results. The cost of probing is approximately 100 times. This aligns with the theoretical expectations of the design, where targeted probing avoids exhaustive path enumeration. Overall, SRmesh achieves a balance between diagnostic precision, resource efficiency, and operational scalability.

V. RELATED WORKS

Active network monitoring. Active measurement techniques have been extensively studied in traditional networks [15], [18]–[20], [22]–[24] and later adapted to virtualized environments to accommodate multi-tenant architectures and overlay networks [16], [17], [21]. With the increasing adoption of RDMA, several recent efforts have focused on designing probe approaches tailored to the RDMA network [42]–[45]. However, SRv6 networks undermine the assumptions of these active probes, making existing solutions insufficient for accurate performance diagnosis.

SR performance measurement. Recent works [30]–[32], [46], [47] have extended active performance measurement protocols to support SR networks. The Simple Two-Way Active Measurement Protocol (STAMP) [48] has been enhanced with SR capabilities in both SR-MPLS and SRv6 data planes [49]. SRv6-PM [50] only focuses on packet loss measurement using alternate marking. Furthermore, SR-STAMP [51] 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 scalable and accurate latency bottleneck diagnosis system tailored for SRv6 networks. First, SRmesh implements deterministic SR-based 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 real network topologies and implement a prototype. The evaluations illustrate that SRmesh achieves high diagnostic accuracy while reducing probe overhead by up to 91.9%.

VII. ACKNOWLEDGMENT

We sincerely thank our shepherd and the anonymous reviewers for their valuable feedback. This work is sponsored in part by the National Natural Science Foundation of China under Grant No. 62394322. Han Zhang is the corresponding author.

REFERENCES

- [1] S. Matsushima, C. Filsfils, Z. Ali, Z. Li, K. Rajaraman, and A. Dhamija, "SRv6 Implementation and Deployment Status," Internet Engineering Task Force, Internet-Draft draft-matsushima-spring-srv6-deployment-status-15, Apr. 2022, work in Progress. [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," 2018.
- [3] Cradlepoint, "Connectivity meets customization with 5g network slicing," 2025, accessed: 2025-05. [Online]. Available: <https://cradlepoint.com/resources/blog/connectivity-meets-customization-with-5g-network-slicing/>
- [4] Telit, "How 5g is reshaping connected health care: A guide," 2025, accessed: 2025-05. [Online]. Available: <https://www.telit.com/blog/5g-connected-health-care-guide/>
- [5] H. Connect, "Why low-latency communication matters for vehicles," 2025, accessed: 2025-05. [Online]. Available: <https://horizonconnect.de/low-latency-communication-connected-vehicles/>
- [6] B. Claise, "Cisco systems netflow services export version 9," Tech. Rep., 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 *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Apr. 2014, pp. 71–85.
- [8] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Rfc1157: Simple network management protocol (snmp)," 1990.
- [9] Y. Lei, Y. Zhou, Y. Lin, M. Xu, and Y. Wang, "Dove: Diagnosis-driven slo violation detection," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, 2021, pp. 1–11.
- [10] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, "Flow event telemetry on programmable data plane," ser. SIGCOMM '20, 2020, p. 76–89.
- [11] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "Pint: Probabilistic in-band network telemetry," ser. SIGCOMM '20, 2020, p. 662–680.
- [12] Q. Huang, H. Sun, P. P. Lee, W. Bai, F. Zhu, and Y. Bao, "Omnimon: Re-architecting network telemetry with resource efficiency and full accuracy," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 404–421.
- [13] K. Yang, S. Long, Q. Shi, Y. Li, Z. Liu, Y. Wu, T. Yang, and Z. Jia, "Sketchint: Empowering int with towersketch for per-flow per-switch measurement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 11, pp. 2876–2894, 2023.
- [14] H. Zheng, C. Tian, T. Yang, H. Lin, C. Liu, Z. Zhang, W. Dou, and G. Chen, "Flymon: enabling on-the-fly task reconfiguration for network measurement," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 486–502.
- [15] Z. Liu, J. Bi, Y. Zhou, Y. Wang, and Y. Lin, "Netvision: Towards network telemetry as a service," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 247–248.
- [16] C. Fang, H. Liu, M. Miao, J. Ye, L. Wang, W. Zhang, D. Kang, B. Lyv, P. Cheng, and J. Chen, "Vtrace: Automatic diagnostic system for persistent packet loss in cloud-scale overlay network," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 31–43.
- [17] S. Zhu, J. Lu, B. Lyu, T. Pan, C. Jia, X. Cheng, D. Kang, Y. Lv, F. Yang, X. Xue *et al.*, "Zoonet: a proactive telemetry system for large-scale cloud networks," in *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, 2022, pp. 321–336.
- [18] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien, "Pingmesh: A large-scale system for data center network latency measurement and analysis," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, p. 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 *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.
- [20] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 419–435.
- [21] A. Roy, D. Bansal, D. Brumley, H. K. Chandrappa, P. Sharma, R. Tewari, B. Arzani, and A. C. Snoeren, "Cloud datacenter sdn monitoring: Experiences and challenges," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 464–470.
- [22] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "NetBouncer: Active device and link failure localization in data center networks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 599–614.
- [23] R. Ding, X. Liu, S. Yang, Q. Huang, B. Xie, R. Sun, Z. Zhang, and B. Cui, "Rd-probe: Scalable monitoring with sufficient coverage in complex datacenter networks," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 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 *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 55–68.
- [25] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," 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)," RFC 7880, Jul. 2016. [Online]. Available: <https://www.rfc-editor.org/info/rfc7880>
- [27] M. A. Vieira, M. S. Castanho, R. D. Pacifico, E. R. Santos, E. P. C. Júnior, and L. F. Vieira, "Fast packet processing with ebp and xdp: Concepts, code, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–36, 2020.
- [28] E. C. Molero, S. Vissicchio, and L. Vanbever, "Fast in-network gray failure detection for isps," ser. SIGCOMM '22, 2022, p. 677–692.
- [29] P. François, C. Filsfils, A. Bashandy, B. Decraene, S. Litkowski *et al.*, "Topology independent fast reroute using segment routing," 2014.
- [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," RFC 9716, Feb. 2025. [Online]. Available: <https://www.rfc-editor.org/info/rfc9716>
- [31] L. Gong, C. Lin, Y. Qiu, and X. Min, "Ping Path Consistency over SRv6," Internet Engineering Task Force, Internet-Draft draft-gong-spring-ping-path-consistency-over-srv6-04, Jan. 2025, work in Progress. [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)," RFC 9259, Jun. 2022. [Online]. Available: <https://www.rfc-editor.org/info/rfc9259>
- [33] C. Li, M. Chen, W. Cheng, R. Gandhi, and Q. Xiong, "Path Computation Element Communication Protocol (PCEP) Extensions for Associated Bidirectional Segment Routing (SR) Paths," Internet Engineering Task Force, Internet-Draft draft-ietf-pce-sr-bidir-path-15, Feb. 2025, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-pce-sr-bidir-path/15/>
- [34] "IP in IP Tunneling," RFC 1853, Oct. 1995. [Online]. Available: <https://www.rfc-editor.org/info/rfc1853>
- [35] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 21–30, 2002.
- [36] T. He, L. Ma, A. Swami, and D. Towsley, *Network tomography: identifiability, measurement design, and network state inference*. Cambridge University Press, 2021.
- [37] S. Amante, J. Rajahalme, B. E. Carpenter, and S. Jiang, "IPv6 Flow Label Specification," RFC 6437, Nov. 2011. [Online]. Available: <https://www.rfc-editor.org/info/rfc6437>
- [38] S. Gay, P. Schaus, and S. Vissicchio, "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," *arXiv preprint arXiv:1710.08665*, 2017.

- [39] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.
- [40] FRRouting Project, "Frrouting: Free range routing suite," <https://frrouting.org/>, 2025, accessed: 2025-05.
- [41] G. Times, "World's first 1.2tb ultra-high-speed backbone network opens in china," 2023, accessed: 2025-05. [Online]. Available: <https://www.globaltimes.cn/page/202311/1301765.shtml>
- [42] K. Liu, Z. Jiang, J. Zhang, S. Guo, X. Zhang, Y. Bai, Y. Dong, F. Luo, Z. Zhang, L. Wang *et al.*, "R-pingmesh: A service-aware roce network monitoring and diagnostic system," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 554–567.
- [43] K. Liu, J. Zhang, Z. Jiang, X. Zhang, S. Guo, Y. Bai, Y. Dong, Z. Zhang, X. Shi, L. Wang *et al.*, "Hostmesh: Monitor and diagnose networks in rail-optimized roce clusters," in *Proceedings of the 8th Asia-Pacific Workshop on Networking*, 2024, pp. 122–128.
- [44] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 202–215.
- [45] T. Ma, T. Ma, Z. Song, J. Li, H. Chang, K. Chen, H. Jiang, and Y. Wu, "X-rdma: Effective rdma middleware in large-scale production environments," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–12.
- [46] R. Gandhi, C. Filsfils, D. Voyer, M. Chen, and B. Janssens, "Performance measurement using twamp light for segment routing networks," *Internet Engineering Task Force, Internet-Draft draft-gandhi-springtwamp-srpm-08*, 2020.
- [47] R. Gandhi, C. Filsfils, D. Voyer, S. Salsano, and M. Chen, "Performance measurement using udp path for segment routing networks," *Internet Engineering Task Force, Internet-Draft draft-gandhi-spring-rfc6374-srpm-udp-03*, 2019.
- [48] G. Mirsky, G. Jun, H. Nydell, and R. F. Foote, "Simple Two-Way Active Measurement Protocol," RFC 8762, Mar. 2020. [Online]. Available: <https://www.rfc-editor.org/info/rfc8762>
- [49] R. Gandhi, C. Filsfils, M. Chen, B. Janssens, and R. F. Foote, "Simple Two-Way Active Measurement Protocol (STAMP) Extensions for Segment Routing Networks," RFC 9503, Oct. 2023. [Online]. Available: <https://www.rfc-editor.org/info/rfc9503>
- [50] P. Loreti, A. Mayer, P. Lungaroni, F. Lombardo, C. Scarpitta, G. Sidoretti, L. Bracciale, M. Ferrari, S. Salsano, A. Abdelsalam *et al.*, "Srv6-pm: A cloud-native architecture for performance monitoring of srv6 networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 611–626, 2021.
- [51] C. Scarpitta, G. Sidoretti, A. Mayer, S. Salsano, A. Abdelsalam, and C. Filsfils, "High performance delay monitoring for srv6 based sd-wans," *IEEE Transactions on Network and Service Management*, 2023.