# Towards Fully-Controllable Packet Steering for AI Backend Networks with SRv6

Shaofeng Wu[1,2], Zhixiong Niu[2], Riff Jiang[3], Guohan Lu[3], Chen Tian[4], Hong Xu[1], Yongqiang Xiong[2]
[1]The Chinese University of Hong Kong, [2]Microsoft Research, [3]Microsoft, [4]Nanjing University

## ABSTRACT

Distributed AI training and inference demand precise traffic control to achieve optimal network performance, yet current traffic management methods remain passive, coarse-grained, and fragmented. We argue that future AI backend optimization requires holistic, proactive, packet-level controllability. In this position paper, we propose a new vision of leveraging Segment Routing over IPv6 (SRv6), a mature WAN technology, to achieve comprehensive traffic controllability in AI backend networks.

We discuss the key advantages of SRv6, including enhanced load balancing, improved failure recovery, and efficient network monitoring. Through our preliminary design and experiments, we demonstrate SRv6's feasibility and potential performance benefits. At the same time, we explicitly outline critical open problems and challenges in adopting SRv6, such as designing effective path assignment algorithms in diverse workloads, scalable control-plane design, high-performance data-plane integration, and effective hardware-software co-design. Furthermore, we identify and present additional open research directions and questions necessary for realizing fully controllable AI backend networks.

## 1 INTRODUCTION

Distributed AI training and inference heavily rely on high-speed inter-node networks (*a.k.a.* AI backend networks) to interconnect GPU nodes, enabling efficient parallelism and supporting ever-increasing model sizes [25, 39, 46, 50]. This backend fabric introduces a range of networking challenges, necessitating careful optimizations to achieve high throughput and low latency.

Existing techniques address specific optimization problems individually, such as load balancing [19, 25], failure recovery [19, 50], network diagnosis [22, 34], and in-network aggregation [26, 30, 41, 44, 50]. For instance, load balancing algorithms like E-ECMP [25], packet spraying [18], and flowlet switching [25] have been proposed to improve upon traditional equal-cost multi-path (ECMP) methods. While these solutions alleviate targeted issues, they remain fragmented without addressing the common fundamental limitations in current traffic control mechanisms.

In this position paper, instead of continuing to optimize each scenario individually, we investigate the foundational question: *What really limits existing traffic control approaches in AI backend networks?* We identify the root issue as the lack of a unified way for *full controllability* that proactively steers each and every packet. Specifically, existing methods exhibit three key limitations:

(1) *Coarse-grained granularity.* Current techniques, such as ECMP, manage traffic at a per-flow granularity, making them ineffective for scenarios requiring precise per-packet path control, including fine-grained load balancing or fast failure recovery.
(2) *Passive traffic management.* Most existing methods lack proactive control and workload-awareness, failing to leverage predictable traffic patterns in distributed AI workloads.
(3) *Poor interoperability.* Individual techniques are tailored to specific scenarios and are hard to integrate or co-tune, complicating overall traffic management.

To address these limitations and provide holistic traffic controllability, we propose Segment Routing over IPv6 (SRv6), a mature WAN technique [8, 10, 11, 13, 16, 24], as the foundational mechanism for AI backend networks. SRv6 enables end-hosts to proactively specify the path of each packet through segment identifiers (SIDs) embedded in packet headers. Compared to other traffic control techniques [28, 37, 40, 49], SRv6 uniquely offers rich programmability, application-awareness, and native IPv6 support, making it an ideal candidate for future AI network infrastructures.

Our initial explorations with SRv6 in AI backend scenarios demonstrate its promise, but also highlight critical challenges in both control-plane scalability and data-plane performance. We discuss potential solutions and outline preliminary designs for enabling the benefits of SRv6-based traffic control.

The contributions of this paper include:

(1) A comprehensive analysis demonstrating why existing traffic control techniques fundamentally limit AI backend optimization (§2.1).
(2) The introduction of SRv6 to AI backend networks, in which we clearly articulate its advantages and demonstrate its suitability (§2.2).
(3) Discussion of key design problems and challenges for integrating SRv6 into existing AI backend infrastructures (§3.1).
(4) A preliminary SRv6-based design (§3.2).
(5) An exploration of open research questions and limitations of existing related approaches regarding AI backend controllability (§4).

## 2 NEED FOR FULL CONTROLLABILITY

In AI backend networks with tens of thousands of GPUs, thousands of switches, and tens of thousands of links [25, 39, 46, 50], effective traffic control is crucial yet challenging due to numerous parallel paths, diverse workloads, and highly dynamic network conditions. In the following, we first analyze the pitfalls of existing traffic control schemes, highlighting their limitations in granularity, responsiveness, and flexibility. Then, we introduce SRv6 as a promising solution, illustrating how its programmability and proactive nature overcome these limitations, paving the way for fully controllable AI backend networks.

### 2.1 Pitfalls of Existing Schemes

We identify three common limitations in how traffic is controlled in AI backend networks.

**Coarse granularity.** Traffic is currently controlled at flow or sub-flow granularity for simplicity and compatibility with RDMA transport [25]. For example, ECMP forwards packets based on 5-tuple hash (or QP-based hash in RoCE v2) and achieves load balancing at per-flow granularity, which however performs poorly for traffic generated by AI workloads due to hash collisions [25, 37, 50]. While packet spraying [18] and adaptive routing [4] make improvements with per-packet load balancing, they do not have precise control on each packet's path and are therefore not applicable for other scenarios that do require such controllability, *e.g.,* in-network aggregation [30, 35, 41]. Most network optimization techniques in AI networks still have to bear with the coarse control at flow level due to infeasibility of manual configuration of routing tables and the lack of efficient per-packet mechanisms [25, 34, 39, 50].

**Passive control.** Inter-node communication in AI backend networks is highly structured, showing predictable patterns and strong correlation with AI workloads. For example, RDMA flows are equal in size for collectives in dense models [19, 25]. However, how traffic travels across the fabric is determined solely by the network for most scenarios and lacks proactive control with awareness of workload patterns and topology changes [25, 39, 50]. For example, message size of a collective communication (CC) operation is determined by the CC library like NCCL [6, 7]. So it is feasible to obtain the sizes of the corresponding RDMA flows before the CC operation actually starts and perform load balancing accordingly. However, both ECMP and packet spraying rely on randomness without assumption on traffic patterns [3, 18, 25, 37], and fail to exploit flow size distribution known in advance.

**Lack of synergy and coordination.** We observe that although different network optimization techniques work well for their specific scenarios, they may be incompatible and even contradicting in terms of traffic control goals. For example, SHARP [26] constructs a spanning tree that defines aggregation hierarchy of computation nodes and switches, and traffic of collective operations strictly follows the pre-defined logical topology and static routing. The deterministic path selection disables adaptive routing [4] since the latter sprays traffic across paths, potentially breaking aggregation hierarchy and yielding hangs or incorrect results. However, adaptive routing is still needed to deal with micro-bursts and load balancing of traffic that do not participate in aggregation in complex AI workloads [44, 50]. The root cause is that there lacks a uniform way of controlling the traffic. When different network optimization goals lead to different traffic control decisions, they have to be configured case-by-case and coordinated carefully.

## 2.2 A Case for SRv6

**SRv6 in WANs.** Segment Routing over IPv6 (SRv6) extends IPv6 by introducing a new Segment Routing Header (SRH), which carries an ordered list of 128-bit Segment Identifiers (SIDs) [8, 13, 24]. Each SID encodes either a topological locator (i.e., "go to this node") or a specific network function or behavior (e.g., steering, service insertion, telemetry sampling). By interpreting the active SID in the SRH, an SRv6-capable node applies the corresponding action and advances the packet to the next segment, enabling path and network service programmability. In addition, SRv6 offers an Optional TLV

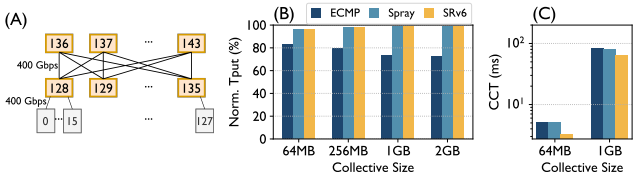


**Figure 1: (A) Example leaf-spine topology with 128 GPUs for simulation [19, 25]. Each GPU is served with a dedicated NIC and link speed is 400Gbps. (B) Normalized throughput of AlltoAll with different collective message size. (C) Collective completion time (CCT) of AlltoAll in logarithm scale with different collective message size under link failure. Link failure happens in the link aggregation group (LAG) between switch 128 and 136 and halves the aggregated bandwidth between these two switches.**

(Type-Length-Value) after segment lists to carry irregular information during packet transmission on a network, providing another aspect of programmability to applications.

Leveraging the three levels of programmability and simplicity of SRv6, major operators have widely adopted SRv6 in their carrier and backbone networks for data center interconnects (DCI), traffic engineering, segment-aware VPNs and 5G UPF chaining [10, 11, 16].

**Potential in AI backend networks.** SRv6 offers us the exact controllability we seek within AI backend networks: traffic can be programmed and managed at the finest granularity over routing layer, *i.e.,* per-packet, allowing each packet to take a unique path and carry unique information from either end-hosts or network. We demonstrate, with two examples, that SRv6 is an ideal data-plane standard for addressing traffic controllability issues of existing techniques and realizing our vision of progressively building a fully controlled AI backend.

*2.2.1 Traffic Load Balancing.* Load balancing heavily determines the performance of AI backend, *i.e.,* whether it can achieve roofline performance with the underlying network hardware and infrastructure when serving bursty, low-entropy and synchronous collective traffic from AI workloads [25, 39, 50]. ECMP, a forwarding strategy widely adopted by data centers to achieve load balancing [28, 37, 49], performs poorly on collective communications driven by RDMA in AI backends due to the lack of 5-tuple randomness and hash polarization [25, 36, 39, 50]. To improve its performance, other load balancing strategies have been proposed for use in AI backends, including QP scaling [25, 39], flowlet switching [20, 25], and packet spraying [18, 50].

SRv6 can help load balancing with its per-packet and host-driven control in three-fold. (1) It is able to support and improve existing techniques considering their specific tradeoffs. We use packet spraying as an example. (2) It is able to support the design of new load balancing algorithms. (3) It is able to achieve fast re-routing under link failures.

**Packet spraying.** Packet spraying works by randomly arbitrating packet paths among equal-cost paths and achieves even load balancing for symmetric topologies [18, 19, 23, 45, 50]. A key factor limiting the deployability of packet spraying is its reliance on proprietary switch hardware [2, 3, 18, 23]. SRv6 facilitates flexible host-based packet spraying as an alternative to switch-based

solutions by allowing hosts to randomly specify the SID of each packet, which aligns with recent calls [18, 50] for standardizing packet spraying in future AI backends.

Another factor limiting the adoption of packet spraying is its poor compatibility with RDMA transport due to out-of-order packet delivery [3, 19, 36]. Recent works [36, 43] require exact path control and path tracing of RDMA traffic to mitigate this issue. They can leverage SRv6 to achieve a simpler and standard design.

**Application-aware load balancing.** Existing load balancing strategies fail to exploit the structural communication patterns of AI workloads. For example, both ECMP and packet spraying rely on randomness without prior knowledge of flow size distribution, while in distributed training , flow sizes are determined before collective starts and can be used to place flows on specific paths in advance to achieve balanced load on these parallel paths. Compared to letting the network distribute packets blindly, this "application-aware load balancing" can potentially avoid packet re-ordering in RDMA, which requires high-end RDMA NIC to solve [3]. The proactive placement of traffic required by application-aware load balancing can be supported gracefully with SRv6. Specifically, a centralized AI scheduler can collect flow size information from collective communication libraries on end-hosts, calculate the optimal flow placement scheme and distribute corresponding SRv6 SIDs to host NICs for packet encapsulation and path control, while the switches simply interpret SRv6 SIDs without other advanced features. We use `astra-sim` [9, 47] and conduct simulations on the leaf-spine topology in Figure 1(A) as an example to demonstrate the benefits of this approach. Figure 1(B) shows that application-aware load balancing enabled by SRv6 can achieve similar normalized bandwidth as packet spraying for a 128-GPU AlltoAll on symmetric leaf-spine topology, without the complexity of multi-path and proprietary hardware.

**Re-routing.** SRv6 is also straight-forward for enabling source-driven re-routing when network failure happens. When the host receives failure signals, *e.g.,* ECN, it can proactively re-route packets by modifying SRv6 SID list. For example, we set the link speed between switch 128 and 136 to 200Gbps in the topology shown in Figure 1(A). Hosts under switch 129-135 can proactively re-route a portion of traffic traveling to switch 128 via switch 136 to other spine switches. We conduct a simulation on this example and the result in Figure 1(C) shows that AlltoAll completion time is reduced by 37.1% and 21.6% for 64MB and 1GB message sizes compared to packet spraying, which performs poorly under topology asymmetry.

*2.2.2 Network Monitoring and Failure Localization.* Distributed AI training is extremely sensitive to network failures, *e.g.,* switch port flapping and link bottleneck [34], due to synchronized communication pattern between GPUs. To prevent single point of network failure from degrading training throughput for a long time, network monitoring and failure localization frameworks are developed to quickly identify the failure type, location and root cause [34, 38]. These frameworks typically use probe packets to traverse links and switches and localize network failures according to joint paths of abnormal probes, *e.g.,* high-RTT ones.

**Enhancing R-Pingmesh with SRv6.** We demonstrate significant improvements to R-Pingmesh [34], a diagnostic framework for AI backend networks, through the adoption of SRv6. Originally, R-Pingmesh exercises possible network paths by varying 5-tuples of probe packets, collectively forming a pinglist. However, due to black-box ECMP hashing on switches, even extensive pinglists cannot guarantee traversal of every path between ToRs or specific desired paths. By explicitly encoding paths in SRv6 packets at end-hosts, SRv6 replaces the pinglist approach with precise path programmability. Our simulations show this dramatically reduces the number of probe packets needed to enumerate all possible paths between ToRs across varying topology scales. Figure 2(A) presents the result, wherein the number of probe packets required to cover all paths is greatly reduced with SRv6's precise path control. This is equivalent to O(TB) reduction of log footprint [27, 34] and $\sim 5\times$ faster detection of failures with the same number of probe packets.

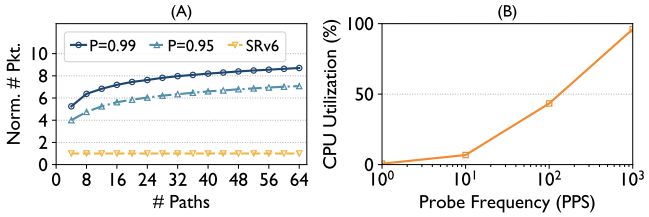


**Figure 2: (A) Number of packets (normalized by number of paths) required at least to probe all paths with a possibility of $P$. $P$ is the possibility of covering all paths using random 5-tuple with ECMP in R-Pingmesh. For SRv6, packet path can be explicitly specified so $N$ paths can always be 100% covered with $N$ packets. (B) Switch CPU utilization with respect to bi-directional `traceroute` frequency.**

Moreover, SRv6 further improves R-Pingmesh by streamlining path tracing. Traditional R-Pingmesh relies on periodic traceroutes to discover paths traversed by probe packets, as 5-tuples alone cannot directly reveal these paths. However, the frequency of traceroutes in R-Pingmesh is restricted to avoid overwhelming switch CPUs with ICMP packet generation, as shown in Figure 2(B). SRv6 inherently preserves path history within its SID stack, allowing for direct extraction from collected probe packets. This eliminates additional traceroute overhead and switch CPU usage, enabling the detection of network failures at microsecond granularity without taxing switch resources.

*2.2.3 Other Usecases.* In addition to load balancing and network diagnosis, we opt for a set of network optimization techniques that can potentially be improved or simplified with better traffic controllability of SRv6. We list these unverified but promising usecases non-exhaustively.

**Micro-burst mitigation.** AI workloads generate bursty traffic [19, 32, 50], which causes microsecond-level congestion even with load balancing strategies that eliminate permanent congestion. Similar to re-routing upon link failures, end-hosts can utilize SRv6 to explicitly specify a new path when they receive congestion signals, *e.g.,* ECN or fine-grained telemetry information, and proactively shift packets to a precise backup path according to topology information.

**Flexible in-network aggregation (INA).** INA offloads vector-sum operations into the network, significantly reducing communication overhead and latency for AllReduce [26, 30, 35, 41] and AlltoAll

in MoE [29, 31, 50]. Existing solutions rely on specialized transport layers [26, 30, 41] or proprietary hardware [26], limiting their compatibility with adaptive routing and other dynamic techniques. SRv6 uniquely integrates path selection and in-network functions by defining an explicit *Aggregate* SID that specifies operation type and aggregation locations. Controllers or hosts can flexibly adjust aggregation topologies, dynamically combining aggregation with basic forwarding functions instead of using static spanning trees [26]. This enables efficient network load balancing and optimal resource usage, ideal for MoE workloads with dynamic and imbalanced communication patterns [29, 31, 50].

Additionally, SRv6 supports unified management of broader in-network operations like multicast and compression from end-hosts transparently [26, 35]. Future hardware adhering to open standards will ensure compatibility and flexibility across evolving AI backend generations [14].

## 3 IS SRV6 A FREE RIDE?

Although we have shown that SRv6 provides promising traffic controllability and potential benefits to multiple usecases in AI backends, it remains unclear how to integrate SRv6 with existing infrastructure and build fully controllable AI backends. A quick analogy is: SRv6 is a set of nice wheels that should not be re-invented, but to build a car, we still need to build or at least upgrade the engine that drives the wheels.

### 3.1 Challenges

We identify major challenges in our initial explorations so far as follows.

*3.1.1 Path assignment algorithm problems and challenges.* Designing path computation algorithms with SRv6 is highly non-trivial. The major complexity is that workload characteristic can heavily affect its corresponding SRv6 path computation strategy. For workloads with a deterministic traffic pattern, *e.g.,* AllReduce with uniform flow size for dense models [19, 25], pre-computing a flow placement scheme based on information from AI scheduler before the job starts is feasible and can achieve corresponding goals, *e.g.,* evenly distributing flows across non-disjoint paths. And SRv6 can be used for deterministic flow placement in this scenario by assigning pre-computed SIDs to hosts to control the path, and the path assignment scheme can remain static throughout runtime. A recent example is Ethereal [19], which assumes that flows have uniform size and let each end-host place local flows evenly across all uplinks in a fully distributed manner. However, emerging workloads often involve semi-predictable traffic pattern due to hybrid parallelism [25, 46, 50], sparsity [32, 50] and multi-tenancy, which prevents path pre-computation. Specifically, this includes:

**Non-deterministic workload traffic.** Workloads may have a non-deterministic traffic pattern. For example, the flow size in AlltoAll of MoE is skewed in both training and inference, and the flow size distribution is dynamic for different micro-batches [31, 32, 50]. In this scenario, even if the collective types can be known in advance, pre-computing a static SID assignment beforehand is either infeasible or sub-optimal. Similar challenges can be seen for pipeline parallelism [46]. A possible solution is predicting traffic distribution and running light-weight path assignment algorithms, but it requires further research on its effectiveness.

**Private workload information.** In the worst case, due to security concerns in public AI backends, the path computation algorithm has no visibility or prior knowledge of the application traffic pattern. In this case, the algorithm needs to be either workload-agnostic or adaptively learn from the ongoing traffic pattern.

**Multi-tenancy.** While a single job's traffic can be deterministic, multiple jobs can share the AI backend and their mixed traffic may not be synchronous, leading to interference. In this case, reserving bandwidth for some tasks may ensure their performance, but also affects the completion time of other jobs. The algorithm needs to coordinate traffic from jobs from different tenants.

*3.1.2 System problems and challenges.* In addition to developing an effective path-selection algorithm to realize our vision, we must also address various system-level issues and challenges to realize the vision.

**Scalable control-plane.** SRv6 requires a separate control-plane to calculate and assign SIDs based on real-time network conditions and job placement in AI backends. While centralized SDN controllers are effective in WAN scenarios with stable topologies [33, 42], the dense connections, dynamic topology, and high flow concurrency in AI backends make centralized SID computation challenging. Frequent real-time events, such as link failures and job updates, can quickly overload a centralized controller in a large AI backend.

**High-performance data-plane.** AI backends rely on RDMA NICs [1, 5] for low-latency and high-throughput communication. Introducing SRv6 adds complexity, requiring NICs to manage topology and path states for SID encapsulation and decapsulation. Current solutions, such as eBPF-based software middle-boxes [12, 48], significantly degrade throughput (68.0Gbps in our tests) and cannot meet RDMA performance demands. Thus, we must develop hardware-based SRv6 data-plane support to maintain line-rate performance.

**Hardware-software co-design.** Effective SRv6 deployment in AI backends also requires hardware-software co-design. Challenges include: (1) how workload information is collected for controller or dataplane — whether by intercepting frameworks such as NCCL or by introducing a proxy where applications explicitly report workload status; and (2) how control-plane and data-plane interact—whether the controller centrally computes and distributes path decisions for each flow, or the data-plane independently computes per-flow routes based on information and commands from the controller.

**Flexible policy expression.** Another challenge lies in the definition and expression of path policy. As AI workloads continuously evolve, we need clear interfaces and expressive mechanisms for path algorithms, enabling network operators to flexibly adjust policies according to changing requirements. This flexibility ensures rapid iteration and adaptation. Moreover, any chosen algorithm must be deployable to ensure consistency and performance.

### 3.2 Preliminary Design

Given existing challenges, we outline a preliminary design for enabling SRv6 in AI backends as shown in Figure 3. We introduce the three major components of our system, together with several auxiliary components.

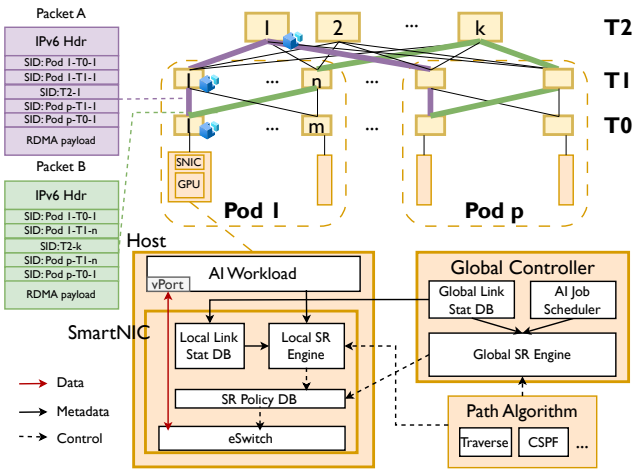


**Figure 3: Design overview of SRv6 integration with AI backends.**

**Path algorithm module**. The path algorithm module implements configurable path-selection strategies. It specifies different algorithms that can be used by both the global and local SR engines to compute SID assignment scheme.

**SR Engine**. The system has two types of SR engines - local SR engine and global SR engine, which are both responsible for calculating SID assignment scheme but operate at different timescales and scenarios.

The global SR engine operates on a coarse timescale and computes the initial SID assignment scheme based on global topology information from global link stat database, *e.g.,* available path and link capacity, and workload information from the AI scheduler, *e.g.,* collective type and its flow size distribution. The global SR engine will initialize SR policy DB, which store active SRv6 policies, on all related hosts with the globally-derived SID assignment scheme. The local link stat DB will also be initialized with information from global stat DB.

The local SR engine is responsible for path computation based on local topology status and application information, and will update the SR policy DB at runtime. Specifically, it can deal with two scenarios: (1) Dynamic events including link failure and job arrival or completion may frequently trigger path computation. (2) AI workloads with non-deterministic traffic pattern, *e.g.,* imbalanced flow size in MoE AlltoAll. The local SR engine perform these path computations locally without frequently triggering a heavy global path computation on global SR engine, which avoids overloading the global controller and ensures scalability. The global controller only needs to periodically perform global computation and updates both local SR policy DB and link stat DB.

**SmartNIC-offloaded data plane**. We facilitate rich SRv6 operations at line rate with SmartNICs on hosts. The global and local SR engine configure the SR policy DB to program the eSwitch on SmartNIC. For RDMA traffic, the eSwitch exposes a virtual port to the AI workload and transparently enforces SRv6 operations based on SID policies at line rate. In addition, the system does not require switches to have advanced features other than being SRv6-capable, *i.e.,* being able to interpret SRv6 SID and functions, which have been supported by SONiC [15, 17], further enhancing the deployability of our preliminary design.

## 4 OPEN QUESTIONS

**What are the switch-level requirements of SRv6?** SRv6 typically forwards packets strictly according to the SIDs. However, it's still an open question how SRv6-enabled switches should handle transient network issues, such as sudden congestion or temporary link failures. For example, should switches proactively adapt the SID-based routing decisions locally, or must they rely solely on centralized controllers to recompute paths? Additionally, deploying SRv6 encapsulation directly at ToR switches could potentially shift path computation entirely onto the network layer, significantly reducing decision-making latency compared to host-level encapsulation. Given that switches inherently possess faster awareness and response to real-time network conditions, we remain open to discussions on the feasibility and implications of ToR-based SRv6 encapsulation and routing adjustments.

**Can we use SRv6 in AI frontend network and cloud datacenter network?** AI frontend is responsible for data ingestion, checkpointing, and logging [25], exhibiting different traffic patterns compared to its backend counterpart. SRv6 can potentially resolve the challenges that frontend networks face, *e.g.,* network heterogeneity and congestion, while managing both frontend and backend in a unified manner. Similarly, cloud networks handle heterogeneous and less predictable workloads, unlike the more predictable aggregated WAN traffic, and tasks tend to be highly heterogeneous. However, for certain cloud workloads such as MapReduce, which involve predictable communication patterns during shuffle phases, applying SRv6 could unlock significant optimization potential.

**How to reduce SRv6 header overhead?** Compared to raw IPv4 or IPv6 packets, SRv6 packets carry additional path information in the SRH, reducing effective payload size. For example, a three-layer CLOS topology path requires 6 SIDs, occupying 96 bytes. While compressed SID encoding (e.g., uSID [21]) generally reduces header size, scenario-specific optimizations—such as sparse SID lists targeting hotspots or adaptively disabling SRv6 for non-incast workloads—also merit further exploration.

## 5 CONCLUSION

We explore leveraging SRv6 as a promising solution to achieve holistic, packet-level traffic controllability in AI backend networks. Our preliminary investigations highlight the feasibility and performance benefits of SRv6. Nevertheless, critical challenges remain in both algorithmic and system aspects. Addressing these open questions and system-level challenges will be essential steps toward realizing fully controllable and optimized AI backend networks.

## REFERENCES

[1] 2021. NVIDIA CONNECTX-6 DX. https://www.nvidia.com/content/dam/en-zz/Solutions/networking/ethernet-adapters/ConnectX-6-Dx-Datasheet.pdf.
[2] 2023. Broadcom AI Interconnect and Tomahawk AI Fabrics. https://techfieldday.com/video/broadcom-ai-interconnect-and-tomahawk-ai-fabrics/.
[3] 2023. To Spray or Not to Spray. https://community.juniper.net/blogs/dmitry-shokarev1/2023/11/21/to-spray-or-not-to-spray?.
[4] 2024. Adaptive Routing. https://docs.nvidia.com/networking/display/ibclusterbringupprocedure/adaptive%2Brouting.
[5] 2024. ConnectX-7 400G Adapters. https://www.nvidia.cn/content/dam/en-zz/Solutions/networking/infiniband/connectx-7-datasheet.pdf.

[6] 2024. NCCL Collective Communication Functions. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/api/colls.html.

[7] 2024. NCCL Environment Variables. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html.

[8] 2024. What is SRv6? https://info.support.huawei.com/info-finder/encyclopedia/en/SRv6.html.

[9] 2025. astra-sim. https://github.com/astra-sim/astra-sim.

[10] 2025. Cisco Supports SoftBank on First Segment Routing IPv6 Deployment in Prep for 5G. https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2019/m02/cisco-supports-softbank-on-first-segment-routing-ipv6-deployment-in-prep-for-5g.html.

[11] 2025. Driving Innovation: Alibaba and Cisco Co-dev SRv6 SONiC Router. https://sonicfoundation.dev/driving-innovation-alibaba-and-cisco-co-dev-srv6-sonic-router/.

[12] 2025. rocev2-srv6. https://github.com/netgroup/rocev2-srv6.

[13] 2025. Segment Routing. https://www.segment-routing.net/.

[14] 2025. SONiC. https://github.com/sonic-net/SONiC.

[15] 2025. SONiC 202505: Powering AI Fabrics and Enterprise Networks with Precision and Insight. https://sonicfoundation.dev/sonic-202505-powering-ai-fabrics-and-enterprise-networks-with-precision-and-insight.

[16] 2025. SRv6: Deployed use-cases. https://blog.apnic.net/2020/05/08/srv6-deployed-use-cases/.

[17] 2025. Static Configuration of SRv6 in SONiC HLD. https://github.com/sonic-net/SONiC/blob/master/doc/srv6/srv6_static_config_hld.md.

[18] 2025. Ultra Ethernet Consortium. https://ultraethernet.org/.

[19] Vamsi Addanki, Prateesh Goyal, Ilias Marinos, and Stefan Schmid. 2025. Ethereal: Divide and Conquer Network Load Balancing in Large-Scale Distributed Training. arXiv:2407.00550 [cs.NI] https://arxiv.org/abs/2407.00550

[20] Peirui Cao, Wenxue Cheng, Shizhen Zhao, and Yongqiang Xiong. 2024. Network Load Balancing with Parallel Flowlets for AI Training Clusters. In *Proc. ACM NAIC*.

[21] Weiqiang Cheng, Clarence Filsfils, Zhenbin Li, Bruno Decraene, and Francois Clad. 2025. Compressed SRv6 Segment List Encoding. RFC 9800. https://doi.org/10.17487/RFC9800

[22] Yangtao Deng, Xiang Shi, Zhuo Jiang, Xingjian Zhang, Lei Zhang, Zhang Zhang, Bo Li, Zuquan Song, Hang Zhu, Gaohong Liu, Fuliang Li, Shuguang Wang, Haibin Lin, Jianxi Ye, and Minlan Yu. 2025. Minder: Faulty Machine Detection for Large-scale Distributed Model Training. In *Proc. USENIX NSDI*.

[23] Advait Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *Proc. IEEE INFOCOM*.

[24] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li. 2021. RFC 8986: Segment Routing over IPv6 (SRv6) Network Programming.

[25] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proc. ACM SIGCOMM*.

[26] Richard L. Graham, Devendar Bureddy, Pak Lui, Hal Rosenstock, Gilad Shainer, Gil Bloch, Dror Goldenerg, Mike Dubman, Sasha Kotchubievsky, Vladimir Koushnir, Lion Levi, Alex Margolin, Tamir Ronen, Alexander Shpiner, Oded Wertheim, and Eitan Zahavi. 2016. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction. In *Proc. COMHPC*.

[27] Chuanxiong Guo. 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proc. ACM SIGCOMM*.

[28] Shuihai Hu, Kai Chen, Haitao Wu, Wei Bai, Chang Lan, Hao Wang, Hongze Zhao, and Chuanxiong Guo. 2015. Explicit path control in commodity data centers: design and applications. In *Proc. USENIX NSDI*.

[29] Chao Jin, Ziheng Jiang, Zhihao Bai, Zheng Zhong, Juncai Liu, Xiang Li, Ningxin Zheng, Xi Wang, Cong Xie, Qi Huang, Wen Heng, Yiyuan Ma, Wenlei Bao, Size Zheng, Yanghua Peng, Haibin Lin, Xuanzhe Liu, Xin Jin, and Xin Liu. 2025. MegaScale-MoE: Large-Scale Communication-Efficient Training of Mixture-of-Experts Models in Production. arXiv:2505.11432 [cs.LG] https://arxiv.org/abs/2505.11432

[30] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. 2021. ATP: In-network Aggregation for Multi-tenant Learning. In *Proc. USENIX NSDI*.

[31] Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. 2023. Accelerating Distributed MoE Training and Inference with Lina. In *Proc. USENIX ATC*.

[32] Wenxue Li, Xiangzhou Liu, Yuxuan Li, Yilun Jin, Han Tian, Zhizhen Zhong, Guyue Liu, Ying Zhang, and Kai Chen. 2024. Understanding Communication Characteristics of Distributed Training. In *Proc. APNet*.

[33] Zhenbin Li, Shuping Peng, Xuesong Geng, and Mahendra Singh Negi. 2025. *PCE Communication Protocol (PCEP) Extensions for Using the PCE as a Central Controller (PCECC) for Segment Routing over IPv6 (SRv6) Segment Identifier (SID) Allocation and Distribution.* Internet-Draft draft-ietf-pce-pcep-extension-pce-controller-srv6-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-pce-pcep-extension-pce-controller-srv6/04/ Work in Progress.

[34] Kefei Liu, Zhuo Jiang, Jiao Zhang, Shixian Guo, Xuan Zhang, Yangyang Bai, Yongbin Dong, Feng Luo, Zhang Zhang, Lei Wang, Xiang Shi, Haohan Xu, Yang Bai, Dongyang Song, Haoran Wei, Bo Li, Yongchen Pan, Tian Pan, and Tao Huang. 2024. R-Pingmesh: A Service-Aware RoCE Network Monitoring and Diagnostic System. In *Proc. ACM SIGCOMM*.

[35] Shuo Liu, Qiaoling Wang, Junyi Zhang, Wenfei Wu, Qinliang Lin, Yao Liu, Meng Xu, Marco Canini, Ray C. C. Cheung, and Jianfei He. 2023. In-Network Aggregation with Transport Transparency for Distributed Training. In *Proc. ACM ASPLOS*.

[36] Xiangzhou Liu, Wenxue Li, and Kai Chen. 2025. Enabling Packet Spraying over Commodity RNICs with In-Network Support. In *Proc. APNet*.

[37] Yadong Liu, Yunming Xiao, Xuan Zhang, Weizhen Dang, Huihui Liu, Xiang Li, Zekun He, Jilong Wang, Aleksandar Kuzmanovic, Ang Chen, and Congcong Miao. 2025. Unlocking ECMP Programmability for Precise Traffic Control. In *Proc. USENIX NSDI*.

[38] Qingkai Meng, Hao Zheng, Zhenhui Zhang, ChonLam Lao, Chengyuan Huang, Baojia Li, Ziyuan Zhu, Hao Lu, Weizhen Dang, Zitong Lin, Weifeng Zhang, Lingfeng Liu, Yuanyuan Gong, Chunzhi He, Xiaoyuan Hu, Yinben Xia, Xiang Li, Zekun He, Yachen Wang, Xianneng Zou, Kun Yang, Gianni Antichi, Guihai Chen, and Chen Tian. 2025. Astral: A Datacenter Infrastructure for Large Language Model Training at Scale. In *Proc. ACM SIGCOMM*.

[39] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proc. ACM SIGCOMM*.

[40] E. Rosen, A. Viswanathan, and R. Callon. 2001. RFC3031: Multiprotocol Label Switching Architecture.

[41] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtarik. 2021. Scaling Distributed Machine Learning with In-Network Aggregation. In *Proc. USENIX NSDI*.

[42] Siva Sivabalan, Clarence Filsfils, Jeff Tantsura, Wim Henderickx, and Jonathan Hardwick. 2019. Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing. RFC 8664. https://doi.org/10.17487/RFC8664

[43] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. 2023. Network Load Balancing with In-network Reordering Support for RDMA. In *Proc. ACM SIGCOMM*.

[44] Haoyu Song. 2024. In-Network AllReduce Optimization with Virtual Aggregation Trees. In *Proc. ACM NAIC*.

[45] Ying Wan, Haoyu Song, Yu Jia, Yunhui Yang, Tao Huang, and Zhikang Chen. 2025. LAPS: Joint Load Balancing and Congestion Control on Unequal-cost Multi-path Data Center Networks. In *Proc. ACM NAIC*.

[46] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Dan Li, Li Chen, Heyang Zhou, Linkang Zheng, Sen Zhang, Yikai Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, Kunling He, Jiaqi Gao, Ennan Zhai, Dennis Cai, and Binzhang Fu. 2025. SimAI: Unifying Architecture Design and Performance Tuning for Large-Scale Large Language Model Training with Scalability and Precision. In *Proc. USENIX NSDI*.

[47] William Won, Taekyung Heo, Saeed Rashidi, Srinivas Sridharan, Sudarshan Srinivasan, and Tushar Krishna. 2023. ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-model Training at Scale. In *Proc. IEEE ISPASS*.

[48] Mathieu Xhonneux, Fabien Duchene, and Olivier Bonaventure. 2018. Leveraging eBPF for programmable network functions with IPv6 segment routing. In *Proc. ACM CoNEXT*.

[49] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. 2021. Hashing Linearity Enables Relative Path Control in Data Centers. In *Proc. USENIX ATC*.

[50] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, and Y.X. Wei. 2025. Insights into DeepSeek-V3: Scaling Challenges and Reflections on Hardware for AI Architectures. In *Proc. ACM ISCA*.