USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

ZooRoute: Enhancing Cloud-Scale Network Reliability via Candidate Path Provisioning and Overlay Proactive Rerouting

Xiaoqing Sun, *Alibaba Cloud;* Xing Li, *Zhejiang University and Alibaba Cloud;* Xionglie Wei, Tian Pan, Ju Zhang, Bowen Yang, Yi Wang, Ye Yang, Yu Qi, Le Yu, Chenhao Jia, Zhanlong Zhang, Xinyu Chen, Xiaobo Xue, Jianyuan Lu, Shize Zhang, Enge Song, and Yang Song, *Alibaba Cloud;* Rong Wen, *Fudan University and Alibaba Cloud;* Biao Lyu, *Alibaba Cloud and Hangzhou Alibaba Cloud Feitian Information Technology and Hangzhou Alibaba Feitian Information Technology;* Yang Xu, *Fudan University;* Shunmin Zhu, *Alibaba Cloud and Hangzhou Feitian Cloud*

https://www.usenix.org/conference/nsdi26/presentation/sun

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ZooRoute: Enhancing Cloud-Scale Network Reliability via Candidate Path Provisioning and Overlay Proactive Rerouting

Xiaoqing Sun[1†], Xing Li[2,1†], Xionglie Wei[1†], Tian Pan[1⋆], Ju Zhang[1], Bowen Yang[1], Yi Wang[1]
Ye Yang[1], Yu Qi[1], Le Yu[1], Chenhao Jia[1], Zhanlong Zhang[1], Xinyu Chen[1], Xiaobo Xue[1], Jianyuan Lu[1]
Shize Zhang[1], Enge Song[1], Yang Song[1], Rong Wen[4,1], Biao Lyu[1,3,5], Yang Xu[4], Shunmin Zhu[1,6⋆]
[1]*Alibaba Cloud* [2]*Zhejiang University* [3]*Hangzhou Alibaba Cloud Feitian Information Technology*
[4]*Fudan University* [5]*Hangzhou Alibaba Feitian Information Technology* [6]*Hangzhou Feitian Cloud*

Abstract

Failures are inevitable in production-scale cloud networks, making reliability a critical concern for both cloud service providers (CSPs) and their tenants. Existing network failure recovery solutions either fail to provide timely failover or require underlay upgrades, forcing tenants to deploy their own high availability systems with additional CapEx & OpEx. However, most tenants lack the expertise or willingness to invest in such systems but are highly sensitive to service disruptions. This motivates CSPs to assume the responsibility of fast and deterministic failure recovery as a cloud service.

In this paper, we present *ZooRoute*, a tenant-transparent, underlay-agnostic network failure recovery service in Alibaba Cloud. ZooRoute utilizes the overlay layer and enables failure bypass by modifying the outer source ports during VXLAN encapsulation. For each destination IP, a set of source port candidates are maintained by proactive probing between tunnel endpoints to guarantee one-shot deterministic traffic reroute onto healthy paths. Scaling such design at planet-scale cloud infrastructures brings challenges such as probing overhead at hypervisors, memory consumption at Tofino gateways, and service disruptions at stateful middleboxes, which we address with a range of novel techniques. Deployed in Alibaba Cloud for 26 months, ZooRoute has significantly improved network reliability, reducing cumulative outage time by 93.19% and masking 98.21% of failures from tenant awareness.

1 Introduction

The rapid growth of cloud computing has fueled a broad shift of applications to public clouds for scalability, flexibility, and cost efficiency [4]. To this end, cloud service providers (CSPs) operate planet-scale infrastructures to host workloads from millions of tenants, where ensuring high reliability is their top priority. However, link- or node-level failures are inevitable at such scale [29–32], leading to service disruptions and tenant losses. Moreover, we observe that, due to a lack of coordination with the underlying infrastructure, tenant-level recovery can surprisingly amplify transient network downtime at the application layer. Although telemetry systems [24, 31, 60, 81] enable CSPs to promptly detect these failures, the subsequent diagnosis and repair still rely on On-Call Engineers (OCEs) and can take minutes to hours, depending on the root causes.

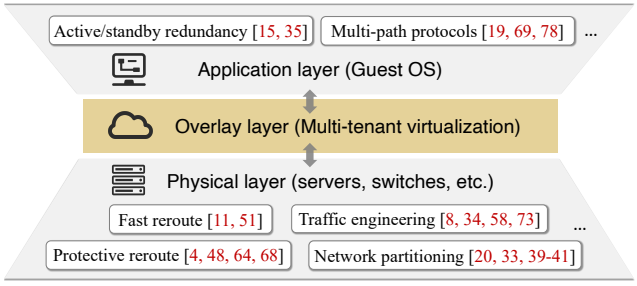


Figure 1: Recovery solutions in public cloud networks.

Existing network recovery solutions struggle to meet stringent server-level agreements (SLAs) or impose burdens on tenants. As Fig. 1 shows, public cloud networks comprise three layers: the physical layer with underlying devices, the overlay layer providing virtual networks for tenants, and the application layer where tenants run their own services. When failures occur, physical-layer solutions either suffer long recovery times [9, 35, 59, 74], rely on underlay upgrades [21, 34, 40–42, 69], intrude on tenant traffic [5, 49, 65] or cannot guarantee deterministic recovery [5, 49, 65, 69]. For instance, traffic engineering can bypass failures globally but takes minutes to act [9, 35, 59, 74]. Protective reroute [69] redirects traffic by modifying the IPv6 Flow Label, requiring a recent kernel version, an IPv6 underlay, and device support for Flow Label hashing. Moreover, protective reroute is opportunistic trial-and-error, without deterministic recovery guarantee. Consequently, tenants are compelled to build their own high availability solutions, most involving redundant resources [16, 36] or protocol stack modifications [20, 70, 79], increasing capital and operating expenses (CapEx & OpEx).

Most tenants lack the expertise or willingness to invest in their own high availability architectures but are sensitive to service disruptions, motivating us to assume the respon-

[†]Co-first authors. [⋆]Co-corresponding authors.

sibility of fast failure recovery as a cloud service. In this work, we propose *ZooRoute*, a tenant-transparent, underlay-agnostic fast network failure recovery service in Alibaba Cloud. ZooRoute provides a unified framework for diverse cloud network scenarios, exploits path diversity for reliability, ensures fast and deterministic recovery, and remains agnostic to heterogeneous underlay devices. Specifically, ZooRoute manipulates the source ports in the "outer" packet headers during overlay encapsulation, which influences Equal Cost Multi-Path (ECMP) hashing in physical switches and routers, thus inducing traffic reroute to alternative paths. To guarantee deterministic recovery, ZooRoute deploys distributed agents on tunnel endpoints (*e.g.*, hypervisor, gateway, middlebox) to proactively probe path status using varying source ports and maintain telemetry records. Available source ports, representing healthy path candidates, will be assigned to tenant traffic immediately to deterministically bypass failures. As operated at the overlay layer, this "path probing, path recording, path switching" mechanism is imperceptible to tenants, agnostic to underlying devices, and supports deterministic global recovery within seconds.

ZooRoute provisions candidate paths to enable deterministic recovery, but doing so at cloud scale introduces significant challenges: **Challenge 1: Massive probed end-hosts vs restricted telemetry budget.** Network-wide telemetry is the basis of candidate path provisioning but full-mesh and high-frequency end-to-end probing in production-scale networks consume substantial CPU cycles. ZooRoute conducts segmented probing at tunnel endpoints, probing only for active IPs/source ports, and source port learning for stateful middleboxes, reducing 95%+ telemetry cost. Besides, it adaptively adjusts probing frequency based on host CPU utilization to avoid resource contention. **Challenge 2: Huge path table vs limited on-chip memory.** Recording source port states is essential for one-shot traffic rerouting onto healthy paths. However, large CSPs often rely on hardware accelerators at critical nodes (*e.g.*, Tofino-based gateway in Alibaba) [14, 26, 48, 51, 53, 55], where on-chip memory is inherently limited. ZooRoute records source ports with priority-based bitmaps and hierarchical indexing, achieving 15x and 25x compression, respectively. Besides, it accelerates table entry updates by three orders of magnitude compared with PCIe via packet-triggered table update, ensuring rapid updates of a vast number of paths when severe failures occur. **Challenge 3: Path switching vs middlebox stateful forwarding.** Path switching may disrupt tenant services, especially for stateful middleboxes (*e.g.*, SLB, NAT) in public clouds. Modifying source ports may trigger reconnections due to missing sessions as traffic is redirected to a different middlebox instance. ZooRoute incorporates a middlebox sensing mechanism that aggregates the source ports corresponding to the same middlebox into the same group. By only selecting source ports within the same group, it ensures that connections are consistently hashed to the same middlebox during path switching.

Our major contributions are summarized as follows:

- We propose ZooRoute, the first overlay-based deterministic network failure recovery service. It is tenant-transparent and enables global failure bypass in large-scale cloud networks (for both intra-region and inter-region) within seconds, without requiring any modification to the network underlay.
- We propose a range of techniques such as on-demand probing, dynamic frequency adjustment, path table compression, packet-triggered table update, and middlebox sensing to reduce telemetry costs, manage path states and prevent unnecessary reconnections, enabling ZooRoute to operate deterministically in production-scale CSP networks.
- During 26 months of deployment in Alibaba Cloud covering all 33 regions, 500k+ servers, and millions of tenants, ZooRoute reduced 93.19% network outage time and masked 98.21% failures without tenant awareness. We share empirical experiences, typical use cases and research insights.

2 Background and Motivation

2.1 Public Cloud Networks

Fig. 2 shows a typical public cloud network with three layers: the physical and overlay layers operated by CSPs, and the tenant-visible application layer, consisting of purchased resources and services (*e.g.*, VM [3], vSwitch[1] [8], vRouter [6], SLB [7]). In the physical layer, CSPs deploy geographically distributed Internet data centers (IDCs) and connect them through private WANs for high-speed communication. For example, Alibaba Cloud operates 90 IDCs across 33 regions, and more than 15% of tenants run cross-region workloads. Multi-path topologies are used within and across IDCs to increase capacity and path diversity, improving reliability [10, 35].

The overlay layer serves as an intermediary enabling tenant traffic to access the shared physical network, providing physical resource virtualization and inter-tenant isolation [39]. Tenant packets are encapsulated with tunnel headers (*e.g.*, VXLAN [50], NVGRE [28]), allowing VMs across regions to communicate in a unified address space, while isolation is enforced via virtual network identifiers (VNIs). At the overlay layer, hypervisors, gateways, and middleboxes act as VXLAN tunnel endpoints (VTEPs), responsible for encapsulating and decapsulating tunnel headers, and forwarding traffic according to VXLAN routing tables and VM-server mapping tables. Specifically, hypervisors, deployed on each server, forward VM traffic [19, 25, 26, 44, 45, 56, 68]. Cloud gateways, located at traffic aggregation points, handle VM-to-VM (intra- or inter-region), VM-to-IDC, and VM-to-Internet traffic [48, 53, 55]. Middleboxes provide network functions like address translation, load balancing, and traffic encryption/decryption [14, 22, 51, 54, 71, 75].

[1]Here, "vSwitch" denotes a tenant-visible logical entity for dividing a VPC into subnets, distinct from the hypervisor vSwitch (*e.g.*, Open vSwitch [56]) used to implement cloud network infrastructure.

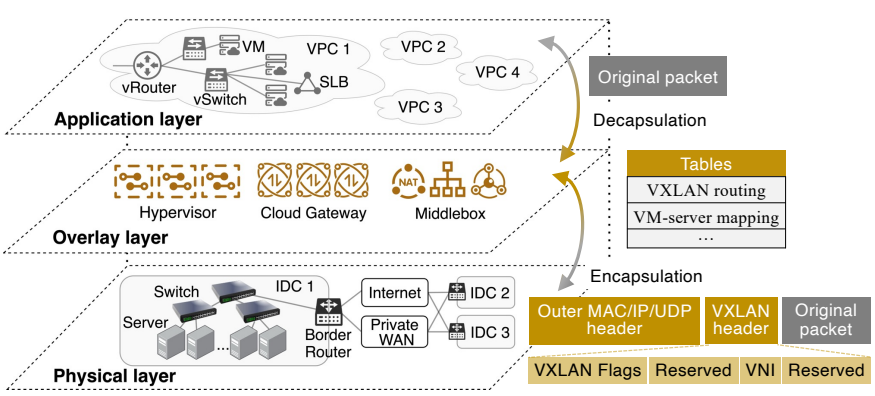


Figure 2: The architecture of public cloud networks (exemplified with VXLAN as the overlay protocol).

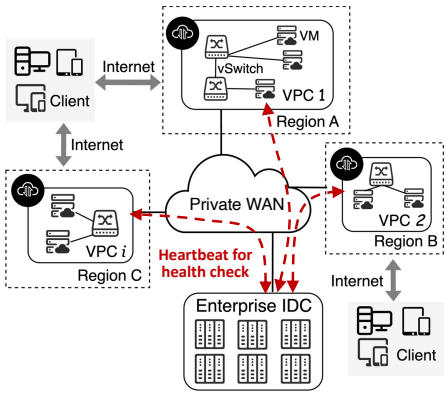


Figure 3: An example of networking SLA monitoring for a gaming company.

2.2 Cloud Network Reliability

Network downtime amplification at tenant applications. As cloud computing evolves, its elasticity and cost benefits has become widely recognized. Latency-sensitive applications like online gaming, financial trading, and video conferencing are moving to public clouds. For backup, data privacy, latency optimization, and regulatory compliance, many tenants adopt multi-site deployments, where workloads communicate both within IDCs and across WANs. As a CSP, we observe that tenant-level failure recovery strategies are often uncoordinated with the provider's infrastructure. As a result, transient network quality degradation can sometimes be amplified at the application layer, severely impacting user experience.

For example, to reduce latency for global players, a gaming company deploys game servers across multiple Alibaba Cloud regions while storing player data in local IDCs for privacy (Fig. 3). The game architecture leverages heartbeat-based monitoring to track server health and uses a primary-backup failover strategy to maintain service continuity. During an 18s cross-region network outage caused by a faulty switch card, three consecutive heartbeat timeouts led the tenant to misidentify its remote server as failed, subsequently triggering a failover. Client connections were forcibly interrupted and reconnected to the backup server during the failover, causing sharp latency spikes and gameplay stutters. Full service recovery took *11min* (much longer than the network downtime), during which over 2,000 players were disconnected due to lost or unsynchronized state, resulting in noticeable operational and reputational impact for the company.

Another example arises in intra-region financial trading, which relies on Aeron, a high-performance transport framework over UDP with userspace ring buffers and NAK-based reliability [1]. When a transient link jitter caused intermittent packet losses for 5*s*, Aeron's sending buffer quickly filled up, forcing the sender to pause transmissions while triggering NAK-driven retransmissions. The retransmission process scans the ring buffer to locate and resend lost packets. Given

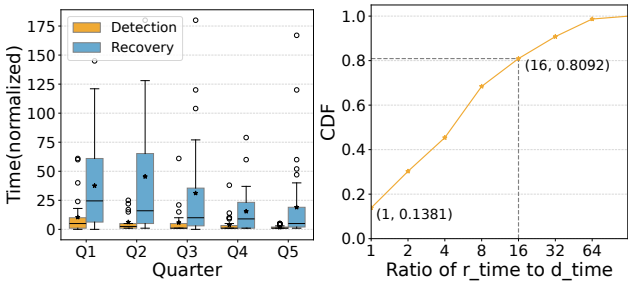


Figure 4: Statistical results of network failure detection time (*d_time*) and recovery time (*r_time*) in Alibaba Cloud.[3]

the high traffic volume, over 70K packets were lost within 5*s*, resulting in repeated pauses, buffer scans, and retransmissions. New packets could not be sent in time and accumulated at the sender, inflating the delivery latency to nearly 1*min*. Such latency amplification imposes tangible financial loss.

Network telemetry fails in prompt failure recovery. Rapid failure detection and recovery are critical to both CSPs and tenants. However, device and link failures are unavoidable in large-scale networks [29–32]. To reduce their impact on tenants, we develop multiple network telemetry systems like VTrace [24] and Zoonet [81] to quickly detect and localize network anomalies. However, they still rely on OCEs to manually diagnose and repair failures given the complex root causes, which usually involve interactions among multiple teams. Sometimes, when the root cause is related to external infrastructure like leased dedicated lines from Internet Service Providers (ISPs), the repair process can take nearly 2 hours.

We investigate the detection time (*d_time*) and recovery time (*r_time*) of 150+ failures in our cloud from 2022-01-01 to 2023-03-31. As shown in Fig. 4 (a), within each quarter, the average *r_time* is more than 3x the average *d_time*, with the peak reaching 10x. By analyzing the outliers in the boxplot, we find that unusually long *d_time* or *r_time* is mainly due to root causes like ISP network failures and switch card

[3]To comply with company policies, data is normalized with *min d_time* as the unit.

exceptions (detailed in Tab A1). We further compute the ratio of *r_time* to *d_time* for each failure and plot a CDF in Fig. 4 (b). We observe that less than 14% of failures are resolved promptly after their detection (*i.e.*, $r\_time \leq d\_time$), while about 20% of failures have long *r_time* exceeding 16x their *d_time*. In summary, with existing telemetry solutions, we can localize failures quickly but cannot ensure timely recovery.

2.3 Network Failure Recovery Solutions

Over the years, numerous approaches have been proposed to repair path failures. However, we observe notable mismatches when directly applying them in our cloud.

Global traffic engineering. Traffic engineering recovers from failures by globally recomputing and rerouting traffic [9,35,59,74]. However, since we deploy a massive number of devices with millions of route entries, a global route reconvergence usually takes at least 10*min*. Some approaches partition the network to reduce the failure domain and thus accelerate route convergence [21,34,40–42]. However, they typically require modifications to the underlying network architecture, such as adding domain SDN controllers.

Local fast reroute. Fast reroute [12,52] enables local repair within seconds by redirecting traffic to backup paths. However, these paths are typically precomputed, which prevents them from handling unexpected failures. In addition, they are provisioned with limited capacity to reduce costs, making them prone to congestion when carrying redirected traffic. Both factors impede full recovery from network outages [69].

Opportunistic protective reroute. To address the limitations of fast reroute, Google proposed protective reroute [58,69], which modifies the IPv6 Flow Label to exploit ECMP path diversity for rapid switchover. Some RDMA failure recovery mechanisms adopt a similar way by modifying the source port to steer traffic to alternative paths [5,49,65]. These solutions, however, are opportunistic trial-and-error: header modifications may trigger path switching but provide no quality guarantee of the new path. Besides, modifying certain protocol fields heavily depends on underlay device support, *e.g.*, Google's private WAN is entirely IPv6-based and all switches are upgraded to support ECMP hashing over Flow Label [69].

Programmable ECMP. To mitigate randomness of ECMP, Tencent proposed programmable ECMP [46], leveraging the ECMP group capability for precise control over hashing. However, this method still relies on the underlying device capabilities (ECMP group is supported on many but not all network devices) and introduces a tight coupling with the hardware.

Resilient overlay routing. Resilient overlay routing improves end-to-end network reliability by dynamically selecting overlay paths to bypass failures, without requiring modifications to the underlying physical network [11,63,72]. However, they require deploying additional nodes at the overlay layer to construct alternative paths and rely on highly complex full-mesh probing to identify the available paths. Besides, overlay paths may incur detours, leading to suboptimal latency.

Tenant-level failure recovery. Due to the limitations of the above infrastructure-level solutions, tenants are compelled to build their own failure recovery solutions. A viable mean is to invest in redundant resources for active/standby replications. Besides, multi-path transport protocols like MPTCP [70] and XQUIC [20,79] improve performance while also enhancing reliability. However, modifying application logic or protocol stack is arduous and costly for tenants, incurring extra CapEx and OpEx. Moreover, as mentioned, tenant-level failure recovery strategies are typically uncoordinated with the CSP's infrastructure, which may amplify the impact of failures.

3 ZooRoute Overview

3.1 Design Principles

Failure recovery as a cloud-native service transparent to tenants. Most public cloud tenants lack the expertise or willingness to invest in their own high availability architectures. A few large tenants build active/standby redundancy at the application layer, but without visibility into underlying failures, their failover often misaligns with the cloud infrastructure, which in turn amplifies the failure. Given tenants' high sensitivity to service disruptions [81], we assume the responsibility of fast failure recovery as a service. The recovery should be transparent to tenants without modifying their traffic so that they benefit from the service without perceiving its presence.

Exploit path diversity for reliability while keeping agnostic to the underlying devices. Multipath connectivity is common within and across IDCs. ECMP is widely enabled on data center switches [76,78] and private WAN routers [35], exploiting path diversity. Moreover, ISPs also extensively enable multipath BGP and ECMP on backbone routers [13,43]. Therefore, Google's protective rerouting is indeed an effective way to leverage path diversity for failure recovery. However, it relies on the Flow Label as the ECMP hash key, which requires an IPv6 underlay and switches capable of Flow Label hashing, resulting in a lengthy rollout due to underlay upgrades [69]. Given the heterogeneity of devices in Alibaba Cloud, we seek failure recovery without modifying the existing underlay.

Fast and deterministic failure recovery. Given that transient network failures can be amplified at tenant applications, and Alibaba Cloud hosts millions of tenants where failures on aggregation links may have a large blast radius [55], we do not rely on opportunistic reroute [5,49,65,69]. Instead, we switch traffic to a deterministically healthy path for recovery.

One failure recovery framework for all cloud network scenarios. Public cloud networks host diverse communication workloads, such as communication between VMs (intra- or inter-region), VMs and the Internet, VMs and local IDCs, as shown in Fig. 5. These scenarios involve heterogeneous devices, including x86 servers, P4 switches, FPGA, *etc*, and face highly concurrent tenant traffic under resource constraints. We aim for the failure recovery framework to generically support all of them, enabling timely end-to-end failure bypass.

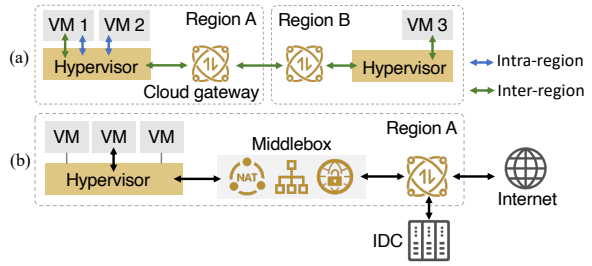


Figure 5: Typical cloud network scenarios: (a) VM-VM [intra-/inter-region], (b) VM-Internet and VM-IDC.

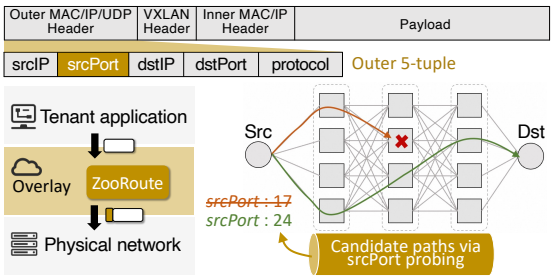


Figure 6: Key idea of ZooRoute.

3.2 Architecture and Workflow

Based on the above design principles, we propose ZooRoute, a fast failure recovery service in Alibaba Cloud.

Tunnel header modification for path switching. Like protective rerouting, ZooRoute modifies packet headers to trigger ECMP hashing in the underlying devices, thereby bypassing network failures. Unlike prior work, this modification occurs at the CSP-owned overlay layer, making it transparent to tenants and agnostic to underlay devices. As shown in Fig. 6, when traffic exits a tenant VM towards the physical network, the VTEP encapsulates it with a VXLAN outer header. The outer source and destination IPs correspond to the servers hosting the source and destination VMs, and the outer destination port is set to 4789 according to the VXLAN standard [50]. The outer source port, however, is undefined and can be set freely. Underlay devices in the physical network are oblivious to inner or outer headers; they perform ECMP hashing solely on the outermost 5-tuple, thereby enabling path switching induced by variations in the outer source port.

Candidate path provisioning for deterministic recovery. To reduce service disruptions, the key distinction of ZooRoute from prior work is deterministic rerouting. Since the hypervisor can steer flows to different paths by changing the outer source port, ZooRoute allows the source VTEP to proactively probe candidate paths (each mapped to a source port) and track their status in real time. When the active path/port shows anomalies, traffic is immediately switched to a pre-probed backup path/port. In practice, ZooRoute conducts round-trip probing between endpoints: forward probes use source ports maintained at the source host, while return probes use source ports maintained at the destination host. A path is deemed unavailable if the RTT exceeds a threshold or a timeout occurs.

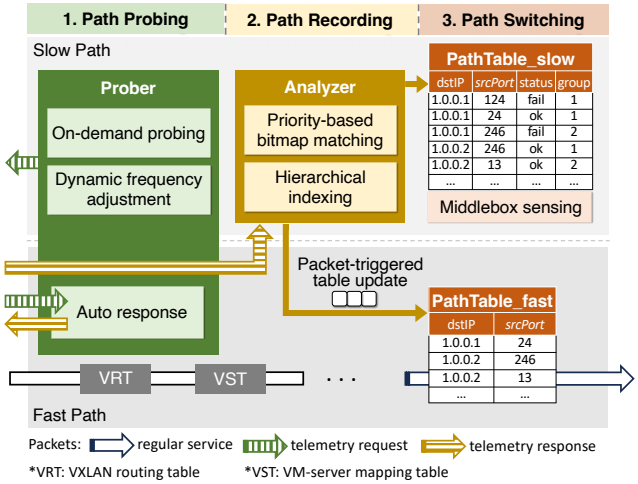


Figure 7: ZooRoute agent architecture.

Segmented probing to cover diverse scenarios. To cover different cloud network scenarios, ZooRoute conducts segmented probing instead of end-to-end probing. It probes under five scenarios with three types of VTEPs, namely, hypervisor (H), middlebox (M), and cloud gateway (G):

- **H-H**: between hypervisors within a region.
- **H-M**: between hypervisors and middleboxes in a region.
- **H-G**: between hypervisors and cloud gateways in a region.
- **M-G**: between middleboxes and cloud gateways in a region.
- **G-G**: between cloud gateways across regions.

Decomposing end-to-end probing into segmented probing reduces the overhead of maintaining candidate paths and enables scenario-specific optimizations (*e.g.*, probe frequency, path storage, stateful vs stateless). By composing segments, ZooRoute achieves coverage over the entire cloud network.

ZooRoute workflow. We design ZooRoute as a distributed system, deployed as agents on VTEPs, to enable fast response and avoid single point of failure. To accommodate heterogeneous VTEPs, ZooRoute agent adopts a unified slow/fast-path architecture. As Fig. 7 shows, the prober, analyzer, and two path tables jointly monitor network status and maintain available paths. The slow path handles probing and path updates, while the fast path utilizes the maintained available paths to support fast rerouting. ZooRoute operates in three steps: probing, recording, and switching. The prober periodically sends telemetry requests to each peer VTEP using different source ports. The analyzer collects the telemetry responses and records metadata (*e.g.*, destination IP, source port, and health status) in `PathTable_slow`. Healthy entries are synchronized to `PathTable_fast`, which is refreshed whenever `PathTable_slow` changes. During tenant traffic encapsulation, the VTEP consults not only the VXLAN routing table and VM-server mapping table to resolve the outer destination IP, but also the path table to select an available source port. Moreover, upon receiving telemetry requests, the ZooRoute agent automatically selects reply paths using PathTable_fast, eliminating slow-path CPU intervention.

3.3 Technical Challenges

VTEP CPU overhead of path probing at cloud-scale. Network-wide telemetry is the basis of path-quality awareness but is costly at large-scale CSPs. We model the telemetry cost per VTEP as $C = (N_v \times N_p \times f) \times c_{process}$, where N_v is the number of peer VTEPs to probe, N_p is the number of source ports to probe, f is the probing frequency, and $c_{process}$ is the processing cost of a telemetry request and its response. At the VTEP level (N_v), scale is the dominant bottleneck. For instance, Alibaba Cloud regions can contain up to 200k servers. Probing all peer VTEPs at a high frequency would incur substantial overhead and impact regular services. At the source port level (N_p), the choice of which and how many ports to probe directly affects ZooRoute's cost and coverage. In principle, topology and hashing information could determine the exact port set that covers all paths to each VTEP, but this is computationally infeasible given the underlay topology scale and dynamics. Heuristically inferring the source-port–to–path mapping via traceroute [37] or In-band Network Telemetry (INT) [38] is also impractical for production, as traceroute strains intermediate devices and INT requires ubiquitous hardware support. Therefore, ZooRoute needs to identify the optimal target VTEPs, source ports, and probing frequency to maximize network visibility within a strict cost budget.

Path recording overhead at Tofino-based cloud gateways. The status of source ports must be recorded to enable one-shot traffic rerouting to healthy paths. However, large CSPs often deploy hardware accelerators to improve performance [14, 26, 48, 51, 53, 55], whose on-chip memory is typically limited. For example, Alibaba builds cloud gateways on Tofino [15]. As a central traffic hub, it must track source port status for tens of thousands of VTEPs. Yet, given Tofino's limited on-chip memory (O(10MB) SRAM/TCAM), maintaining VXLAN routing and VM-server mapping tables is already challenging, let alone adding another large candidate path table.

Unlike the relatively infrequent updates to the VXLAN routing table or VM-server mapping table, network failures at traffic aggregation points require immediate updates to a large number of entries on the path table. Our stress tests show that the maximum throughput from the slow-path CPU to the programmable ASIC over PCIe is about 10k transactions per second. Assuming a gateway maintains paths for 50k VTEPs[4] and, in the worst case, all $N_p = 32$ source ports per VTEP must be updated, the process could take nearly $3min$, during which tenants' traffic may be disrupted by stale path information.

Connection disruptions at stateful middleboxes during path switching. Path switching is vital for fast recovery, but may disrupt normal transmission and cause packet reordering. Some packet reordering is acceptable during failures, and modern kernels plus techniques like reordering buffers [61] make mild reordering manageable for stateless forwarding (see Table A2). Stateful forwarding is more challenging. In many cloud services (*e.g.*, NAT, SLB), changing the source port can break session affinity and trigger connection resets, adding extra latency. Specifically, for scalability and robustness, large CSPs typically deploy clusters of cloud gateways or middlebox instances (*e.g.*, the SLB in Fig. 14) that advertise a virtual IP (vIP) via a Linux virtual server [77] switch (LVS switch, LSW) and distribute incoming traffic using ECMP. As a result, when a flow from src_i to vIP_j changes its source port, it may be redirected to a different middlebox instance. While this has no impact on stateless forwarding, stateful services such as SLB will reset the connection because the corresponding session does not exist on the new instance.

[4] Production data in our cloud shows that in the largest region, the number of VTEPs concurrently communicating with a gateway is close to 50k.

4 Path Probing

4.1 On-Demand Probing

Active VTEP probing. Full-mesh telemetry is redundant due to traffic sparsity. In Alibaba Cloud, over 99% of VTEPs communicate with less than 25% of their regional counterparts. Consequently, ZooRoute initiates a probing task on VTEP_i to VTEP_j only if there is tenant traffic between them, which can be detected at VTEP_i during packet encapsulation, and the task will be released after a duration of inactivity.

Active source port probing. ZooRoute supports two probing modes: full-probe and partial-probe. In full-probe mode, ZooRoute randomly probes N_p source ports, starting from 10,000 to avoid conflicts with well-known or reserved ports. The value of N_p is tuned based on the CPU usage distribution of the destination VTEP (detailed in Fig. A2). Empirically, ZooRoute set $N_p = 32$ to balance telemetry cost and the risk of traffic concentrating on a single underlay path. For cross-region scenarios, where cloud gateways handle higher traffic volumes, we increase N_p to 64 to ensure sufficient entropy. Besides, to avoid path scarcity, if more than half of the ports on a VTEP are unavailable, ZooRoute selects an additional N_p ports for probing. In partial-probe mode, ZooRoute only probes ports actively carrying tenant traffic. By default, intra-region traffic uses partial probing to minimize cost; however, upon detecting anomalies (*i.e.*, when the path associated with the current source port becomes unhealthy), it dynamically switches to full probing to rapidly identify healthy alternatives. Conversely, inter-region (G-G) traffic always employs full probing due to the smaller number of endpoints and the larger failure blast radius as traffic aggregates at gateways.

Source port learning in stateful forwarding. ZooRoute conducts round-trip probing to to verify bidirectional path availability. Namely, while source VTEPs actively select source ports, destination VTEPs must also maintain their own source port lists to direct return traffic. However, middleboxes do not initiate traffic themselves, which makes the active probing unnecessary. ZooRoute thus implements a passive learning mechanism for these middleboxes. As shown in Fig. 8, since stateful middleboxes maintain session tables that record the

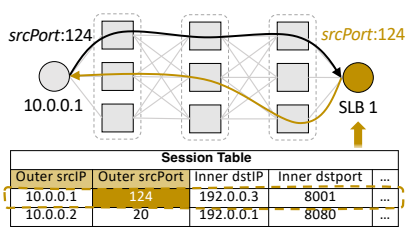


Figure 8: Source port learning in stateful forwarding scenarios like SLB.

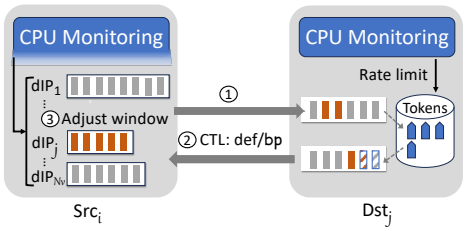


Figure 9: CPU-based frequency adjustment for source and destination VTEPs.

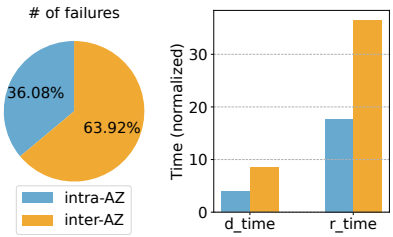


Figure 10: Scenario distribution of network failures in Alibaba Cloud.

outer header information of incoming packets, it can passively select return paths using the source ports employed by peers. For example, when server 10.0.0.1 accesses backend service 192.0.0.3 via a load balancing service (SLB 1), the session table of SLB 1 records both the inner 5-tuple, outer source IP and source port 124. Later, when SLB 1 sends packets back to 10.0.0.1, it queries the session table and mirrors 124 as its own outer source port for an available path. If that return path fails, the source VTEP can correctly infer the incident due to the absence of telemetry responses and proactively switches to another source port; SLB 1 subsequently adopts this new port for its responses, effectively bypassing the failure. This design enables middleboxes to rely on the existing session table, eliminating the need for active probing.

4.2 Dynamic Frequency Adjustment

CPU is the primary bottleneck for VTEPs during path probing. As shown in Fig. 9, ZooRoute monitors CPU utilization per VTEP, which reflects traffic processing load [47], and uses a per-destination packet sending window at the source to adapt probing frequency. The source-side cost mainly comes from constructing and sending telemetry requests. To avoid impacting regular service, ZooRoute applies exponential backoff when more than 7 of the last 10 CPU utilization samples exceed 80% and the probing rate $\geq 30kpps$. Conversely, if all of the last 10 samples are below 70%, it restores the default rate of one probe per second per source port. To stay effective under reduced frequency, ZooRoute prioritizes telemetry tasks. Fig. 10 summarizes incidents in Alibaba Cloud from 2022.1.1 to 2023.3.31, showing that ~64% of failures affect inter-AZ communication, with outages lasting more than twice as long as intra-AZ failures. Therefore, when under tight resources, ZooRoute prioritizes inter-AZ probes, which are identified via the TTL in negotiation response packets.

At the destination, concurrent probe packets from multiple sources may converge into an incast [17], creating processing pressure. ZooRoute employs a token bucket for rate limiting and signals the sources to adjust its sending window W through the CTL field in telemetry response packets (Fig. 9). Specifically, the destination server dst_j adjusts the maximum limit of its token bucket based on CPU utilization. By default, it responds to each source with CTL $= def$. When tokens are insufficient, the response carries a backpressure signal (CTL $= bp$), instructing source src_i to lower its telemetry frequency. As shown in Algorithm 1, src_i by default increases W_j additively. Upon receiving backpressure signals, src_i reduces W_j based on the number of responses carrying CTL $= bp$ (N_{bp}) and the number of lost telemetry requests (N_l).

Algorithm 1: Telemetry frequency adjustment at the source VETP with CTL.

Require: CTL.value, W_j, N_{bp}, N_l
if CLT.value == *def* **then**
  $W_j = W_j + 1$
  return probe dst_j with sending window W_j
else if CLT.value == *bp* **then**
  $W_j = W_j - N_{bp} - N_l$
  if $W_j > 0$ **then**
    return probe dst_j with sending window W_j
  else
    return probe dst_j every $2^{(1-W_j)}$ seconds
  end if
end if

5 Path Recording

5.1 Path Table Compression

To compress path table at production scale, ZooRoute does not store a set of complete source ports (16-bit each) for each Dst VTEP. Intead, it designates a base source port (*e.g.*, 10,000) and uses a bitmap to represent the set of source ports for each Dst VTEP, where "1" indicates available and "0" indiates unavailable (Fig. 11). While this data structure suffices for x86-based VTEPs, Tofino-based gateways face strict limits on programmability and on-chip memory. Therefore, we further tailor scenario-specific compression based on peer VTEP scales: Priority-based Bitmap Matching (PBM) handles pipeline constraints of Tofino in cross-region (G-G) scenarios, while Hierarchical Indexing (HI) minimizes memory footprint for massive intra-region (H-G) endpoints.

Priority-based bitmap matching (PBM). After compressing the path table, ZooRoute selects an available path by randomly choosing a source port whose bit is "1" in the destination IP's bitmap. Implementing such random selection on Tofino is non-trivial because the pipeline lacks loop support. To address this, ZooRoute conducts random port selection using the packet's inner 5-tuple hashing and priority-based bitmap matching. Specifically, we construct a priority table with N_p groups, each containing N_p masks whose length equals the bitmap (*i.e.*, N_p).

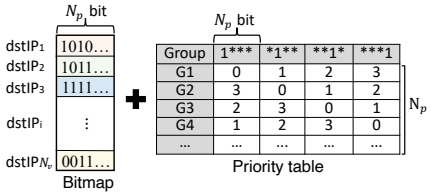


Figure 11: Path table compression with PBM for cross-region (G-G) scenarios.

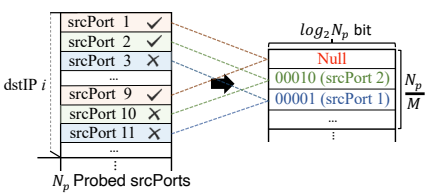


Figure 12: Path table compression with HI for intra-region (H-G) scenarios.

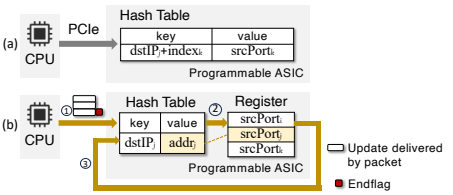


Figure 13: Path table updates delivered via (a) PCIe or (b) packets.

As Fig. 11 shows, with $N_p = 4$, a group holds masks "1***", "*1**", "**1*", and "***1". Within each group, masks are assigned priorities via circular shifts, with 0 as the highest priority. This table is shared across all destination IPs. When sending a packet, ZooRoute hashes its inner 5-tuple to a group, and matches masks against the destination's bitmap in priority order. The source port of the first matching mask is then selected for VXLAN encapsulation.

For example, consider $dstIP_2$ in Fig. 11, where ZooRoute probes source ports 10001-10004, yielding a bitmap of "1011" (*i.e.*, port 10002 is unavailable). A packet hashed by its inner 5-tuple to group G3 finds that the top-priority mask "**1*" successfully matches the bitmap. Consequently, it uses source port 10003 for VXLAN encapsulation. If source port 10003 fails, the bitmap becomes "1001", and mask "**1*" no longer matches. ZooRoute then scans G3 in priority order, selecting the next matching mask "***1", so the packet uses source port 10004 for VXLAN encapsulation at this time.

Hierarchical indexing (HI). In intra-region (H-G) scenarios, gateways serve as traffic hubs and must track the availability of source ports for tens of thousands of VTEPs in the region. ZooRoute introduces hierarchical indexing for further path table compression. As shown in Fig. 12, instead of recording status for all the probed N_p source ports of $dstIP_i$, ZooRoute groups every M source ports into one entry in the path table, reducing entries to N_p/M. In implementation, the M source ports are selected at a fixed stride of N_p/M. For example, in Fig. 12, with $N_p = 32$ and $M = 4$, source port 1, 9, 17...are grouped together. The table entries are filled as follows:

- If all source ports in this group work well, the corresponding entry will remain empty (as depicted in red).
- If there exist "bad" source ports, the entry will be randomly filled with a working source port in the same group (as depicted in green).
- If all source ports are unavailable, a working one from other groups will be randomly chosen (as depicted in blue).

When encapsulating a packet, ZooRoute computes the path table index for a group of source ports. If the entry is empty, it randomly selects one of the M source ports; otherwise, it uses the recorded one. In theory, HI requires $O(\frac{M}{log_2 N_p})$ times less memory than PBM. The tradeoff is that in the worst case (*i.e.*, exactly one "bad" port in each group), HI exposes $M-1$ fewer usable ports than PBM. Such corner cases are rare. Besides, compared to cross-region G-G traffic, intra-region H-G traffic is lighter, thus less prone to polarization. In practice, we set $M = 8$ to balance memory efficiency and polarization risk.

5.2 Packet-Triggered Table Updates

Failures at traffic aggregation points can trigger massive path updates. A naive design stores destination IPs and their source ports as {key, value} pairs in on-chip hash table, relying on the CPU to push updates via PCIe (Fig. 13(a)), but PCIe bandwidth limits update concurrency. ZooRoute instead uses an in-network update mechanism. As shown in Fig. 13(b), it combines hash tables and registers on Tofino: registers store frequently changing source port status (read/write at line rate during packet processing), while the relatively static mapping from destination IPs to registers resides in the hash table.

When a path state change is detected, the CPU sends negotiation packets carrying special PKT markers (see Appendix A) (①). Upon reception, Tofino extracts the destination IP and source port fields from the packet header and updates the corresponding entry (②). Since Tofino supports Tbps throughput, the bottleneck now shifts to the CPU's ability to construct and send these packets. To further accelerate updates and reduce bandwidth, ZooRoute batches multiple updates per packet. After each update, Tofino invalidates the processed header and re-injects the packet via loopback to handle the next update, repeating until an EndFlag is reached (③).

6 Path Switching

Changing the source port can rehash flows to a different middlebox instance, causing session loss and connection resets. A strawman mitigation is to synchronize session state across instances. For example, Alibaba Cloud syncs all sessions every 30*s* and incrementally syncs new connections after the three-way handshake. While covering most cases, this method misses traffic with incomplete handshakes or unsynchronized sessions. ZooRoute arms stateful forwarding with middlebox sensing. By adding a "Backend ID" field in probe packets (see Appendix A), it ensures traffic can be steered to the original middlebox instance after a path switch. Specifically, when a middlebox returns a telemetry response, it fills this field with its MAC address. The source VTEP then groups source ports by Backend ID. As Fig. 14 illustrates, source ports 24, 124, and 12 are grouped together since they receive responses from the same middlebox instance (same Backend ID).

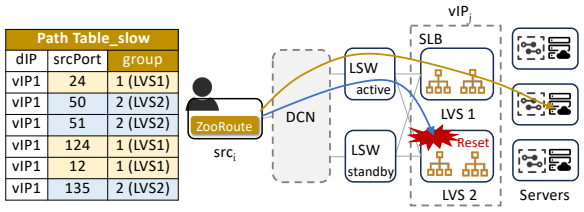


Figure 14: Path switching in stateful forwarding scenarios like SLB w/wo middlebox sensing.

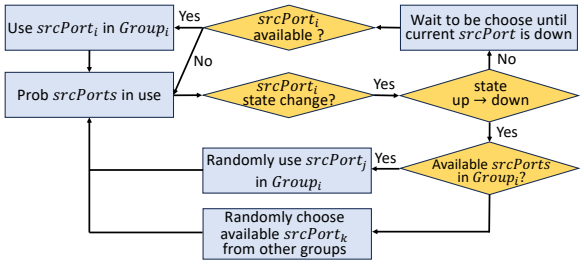


Figure 15: Process of path switching with middlebox sensing.

Fig. 15 illustrates the strategy. For new connections traversing stateful middleboxes, the source VTEP randomly selects one source port per group to balance load across instances. If an in-use port $srcPort_i$ becomes unavailable, ZooRoute preferentially replaces it with $srcPort_j$ in the same group $Group_i$, keeping traffic on the same instance to avoid connection resets. If all ports in $Group_i$ are unavailable, likely indicating an instance failure, it selects a port from a different group, rerouting affected traffic to another instance. For stability, ZooRoute changes paths only when the currently used source ports fail; it does not proactively switch paths when group membership changes or a previously unavailable port recovers. Moreover, since middlebox sensing is incompatible with path table compression, ZooRoute skips compression for stateful middlebox vIPs. As such vIPs are few, the memory impact is negligible.

7 Evaluation

To evaluate the effectiveness of ZooRoute's optimizations in path probing, recording, and switching, we collect data from three representative regions of varying sizes: a large region wtih 100k+ VTEPs, a medium region with 50k+ VTEPs, and a small region with 10k+ VTEPs. For now, ZooRoute has been deployed in production for 26 months, covering diverse cloud networking scenarios across 33 regions in Alibaba Cloud. We report ZooRoute's online overheads and performance, and illustrate its effects and limitations with three typical cases.

7.1 Evaluation of Optimizations

Costs in path probing. We measure telemetry packet counts over a 30-*min* period in the three regions. Taking full-mesh probing with $N_p = 32$ as the baseline, Fig. 16 (a) shows the

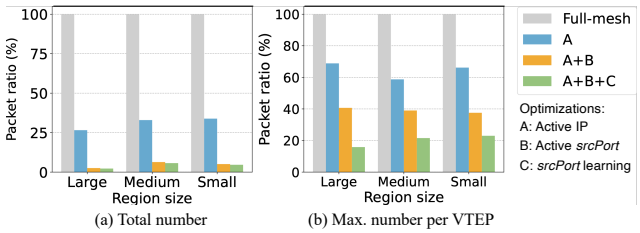


Figure 16: Remaining probing packets after optimizations: (a) total and (b) per-VTEP Max. count relative to full-mesh.

proportion of total telemetry packets remaining after a series of optimizations. Fig. 16 (b) highlights the reduction in the maximum number of packets per VTEP, shown as a ratio to the maximum count observed in the full-mesh baseline. Our results show that active IP probing reduces telemetry costs by 60%~70%. This strategy is more effective in larger regions, likely due to a sparser communication matrix among VTEPs. On this basis, active source port probing further brings the overall cost down to around 5%, with the exact reduction depending on the specific traffic characteristics of each region. While the source port learning mechanism has a negligible impact on overall regional costs, it is particularly beneficial for stateful middleboxes, which typically communicate with a high volume of VTEPs. By directly exempting them from path probing, this strategy lowers the VTEP-level maximum telemetry packet count in their regions.

Moreover, our monitoring of VTEP's CPU usage in Alibaba Cloud shows that less than 1‰ of VTEPs have their 99th percentile (P99) CPU utilization exceeding 80% per day. The dynamic frequency adjustment mechanism is thus a last-resort solution, triggered only in extreme cases to prevent resource preemption, with minimal impact on ZooRoute's agility.

Path table storage and updates. We set the baseline as storing one available source port in every entry of P4 gateway's built-in hash table. Because this design is infeasible to compile at production scale, it serves only as a theoretical reference. Fig. 17 depicts the memory usage of ZooRoute's path table after compression with PBM or HI, shown as a ratio to the baseline. PBM cuts memory usage by ~15x, while HI achieves ~25x compression. Discrepancies across regions result from varying fractions of vIPs associated with stateful middleboxes. Entries for such vIPs remain uncompressed to retain grouping information.

Production data shows that, in the largest region, a single gateway concurrently serves nearly 50k VTEPs. We therefore emulate a substantial outage, where each of the 50k VTEPs must update state for N_p=32 source ports. Fig. 18 presents the completion time under different strategies. Compared to PCIe transmission, packet-triggered messaging accelerates table updates by three orders of magnitude. Besides, when using PCIe, HI takes ~5x longer than PBM, because HI needs to record one available source port for every group. Nevertheless, with packet-triggered messaging, this gap becomes negligible.

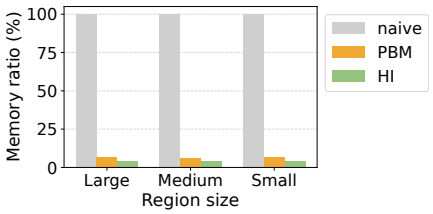


Figure 17: Path table memory usage after compression (relative to naive baseline).

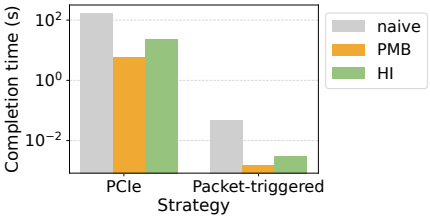


Figure 18: Path table update completion time under different messaging strategies.

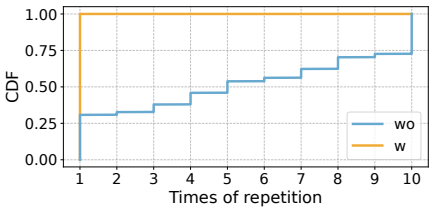


Figure 19: Cumulative distribution of repeat times w/wo middlebox sensing.

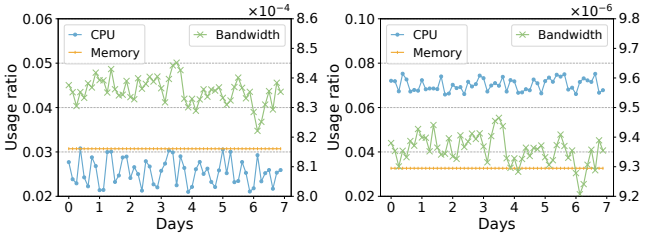


Figure 20: P99 resource usage of ZooRoute on x86 servers (left) and Tofino-based gateways (right).

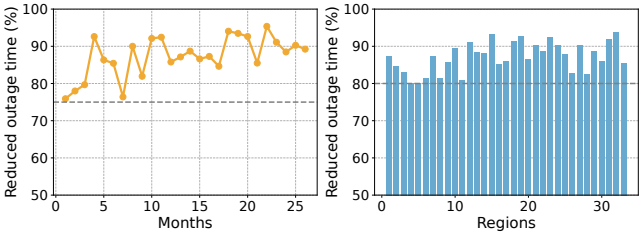


Figure 21: Outage time reduced by ZooRoute over months (left) and across regions (right).

Stateful forwarding with middlebox sensing. We deploy an SLB with vIP_1 (running on three LVS instances) in Alibaba Cloud, and set up a basic HTTP service on the backend VM. A client VM issues requests to vIP_1 using CURL [2]; upon failures it retries until success with a cap of 10 times. We repeat this process for 1,000 rounds. To emulate ZooRoute's behavior during failures, the hypervisor reassigns different source ports every three packets within the same flow. Fig. 19 plots the CDF of per-round retry counts, with and without middlebox sensing. It is evident that, originally, about 25% of rounds hit the retry cap and still fail; with middlebox sensing, all requests succeed on the first attempt. This is because after changing source ports, packets of the same flow may be steered to different LVS nodes that lack the corresponding session states, thus unable to complete the TCP three-way handshake and establish a connection. The middlebox sensing strategy confines changes within the set of source ports mapped to the same LVS, preserving session continuity.

7.2 Overhead of ZooRoute

We assess ZooRoute's online overhead with the resource consumption across VTEPs in the Large region. Fig. 20 reports a week of daily P99 resource usage on x86-based servers and on Tofino-based gateways. On servers, ZooRoute uses 2%~3% CPU, 3% memory, and <0.085% NIC bandwidth. Memory usage appears fixed because we pre-allocate ZooRoute's memory to its maximum footprint to avoid interference with tenant workloads. Besides, with dynamic frequency control, ZooRoute backs off under resource pressure and does not compete with regular services. On Tofino-based gateways, ZooRoute consumes <7.5% CPU, 3% on-chip SRAM, and <0.001% of bandwidth. The seemingly higher CPU stems from the fact that packets are directly processed by Tofino, while the CPU only performs auxiliary tasks like configuration management. Overall, these overheads are modest and acceptable for large CSPs, especially compared to the reputational and revenue risks of SLA violations.

Table 1: Top 3 failure types mitigate by ZooRoute.

Root causes	Scene	Avg. RT	Avg. RR
Jitter in long-haul link	inter-region	7.47*s*	84.67%
Switch card or port anomaly	intra-region	139.52*s*	93.12%
Fiber issue or cutover in ISP	inter-region	295.33*s*	98.60%

* RT: reduced outage time; RR: outage-time reduction ratio.

7.3 Online Performance

ZooRoute has been deployed in Alibaba Cloud for 26 months. Over this period, 46.6% of failures are intra-region, mainly caused by device faults like switch port anomalies, while the remaining are inter-region, mostly due to link issues like long-haul link jitter. To quantify the benefit, we derive a baseline outage time from ZooRoute's probing of faulty source ports, and measure the actual end-to-end impact using Zoonet [81], a proactive overlay telemetry system. By comparing their loss windows, we find that ZooRoute cumulatively reduces 93.19% of outage time, masking 98.21% of failures from tenants awareness. As shown in Fig. 21, ZooRoute consistently provides large outage reductions, with monthly ratios above 75% and regional ratios exceed 80%. Variations are caused by differences in failure types and root causes.

We group incidents by root cause and report the Top 3 types with the largest cumulative outage reductions by ZooRoute in Table 1. *i*) Transient jitters on long-haul links are common, usually last 5s~15s and harm latency-sensitive applications like live streaming and financial trading. ZooRoute bypasses them within 1*s*, typically without tenant packet loss. *ii*) Switch card/port anomalies occur sometimes, with duration from sec-

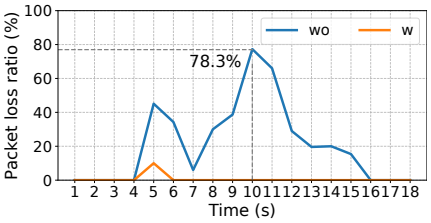


Figure 22: Packet loss w/wo ZooRoute during a long-hual link jitter.

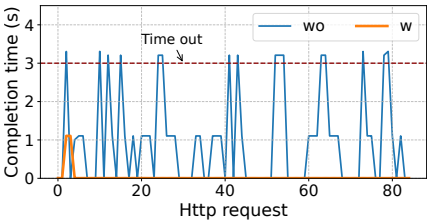


Figure 23: HTTP request latency w/wo ZooRoute during a switch card failure.

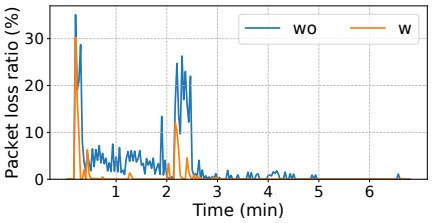


Figure 24: Packet loss w/wo ZooRoute when two border routers down.

onds to minutes; most are mitigated by ZooRoute within 3*s*. *iii*) ISP fiber cuts are rarer but can persist for hours and lie beyond CSP's control; ZooRoute reduces their impact and facilitates cross-enterprise coordination. Next, we present three cases to illustrate ZooRoute's effectiveness and limitations.

Case 1: Jitters on a long-haul link. A long-haul link between two regions fluctuated for 12*s*. As shown in Fig. 22, the loss rate of probe packets peaked at 78.3% at the 6th second. With ZooRoute, tenant traffic was rerouted to a stable link within 1*s*, with transient loss below 10% before full recovery.

Case 2: Line-card failure on a switch. A line card on a distribute switch in AZ_A failed for 2*min*, affecting 67% of inter-AZ traffic. We compare two VMs that communicated with an impacted VTEP: VM1 from AZ_B (without ZooRoute), and VM2 from AZ_C (with ZooRoute). Fig. 23 plots the completion time of their HTTP request latency. During the incident, VM1 experienced repeated retries and timeouts. In contrast, VM2 saw only a short latency spike due to initial retries and remained unaffected thereafter.

Case 3: Half of border routers failed. A program bug caused two of four border routers to fail in sequence. Fig. 24 shows inter-region packet loss rising to 35% initially and 26% after 2*min*, lasting 4.5*min* with residuals until 6*min*. ZooRoute substantially reduced loss but could not recover within 3*s*. This is because: i) unstable control/data-plane during ISIS convergence across many ports, triggering frequent re-connections and inconsistent ECMP hashing; source ports chosen from historical probes may be steered to defunct paths. ii) A 50% capacity drop left remaining links near saturation; redirecting impacted traffic risked exceeding 100% utilization and inducing loss. Making ZooRoute more robust under unstable and highly utilized networks is a key direction for future work.

8 Experiences

The value of deterministic failover in public clouds. Some prior work [5, 49, 65, 69] employs opportunistic rerouting to bypass failures. In contrast, ZooRoute performs deterministic rerouting by shifting traffic to candidate paths that have been proactively probed and validated as healthy, enabling a one-shot switchover. This property is crucial in public clouds, because as illustrated in §2, transient network degradation can be amplified by applications and harm user experience. We study the cross-region scenario, where path redundancy is typically limited. Statistics show that in cross-region failures over the last six months, the average proportion of remaining available ports on impacted gateways was 79.52%, with a minimum of 24.22%. Let p denote the probability that a randomly chosen port is healthy. Assuming independent port selection in both directions, the success probability of a bidirectional probe is p^2, and the number of retries T follows a geometric distribution with $E[T] = \frac{1}{p^2}$ [46]. This yields an average of 1.58 attempts, with extreme cases requiring 17.05 retries. When the remaining port ratio falls below 70%, more than two retries are expected. These observations quantify the long-tail risk and motivate deterministic failover.

Balance sensitivity and stability in path switching. Though modern kernels and reordering buffers [61] can tolerate mild packet reordering (see Table A2), minimizing route flapping is still essential for network stability. We find that switching paths on a single probe loss is overly aggressive and injects avoidable churn. ZooRoute therefore applies hysteresis to filter transient jitter: upon a probe failure on a source port, it immediately sends a second probe, and marks the port as bad only if both probes fail. Besides, to handle intermittent packet loss, ZooRoute further adopts a 3-out-of-5 policy, declaring a port unhealthy when three of the last five probes fail. Beyond binary up/down states, ZooRoute uses historical telemetry to guide path selection. Since our data indicates that recently recovered ports carry a higher risk of recidivism, ZooRoute ranks candidate ports by their failure-free duration and favors those with longer stable histories. To avoid concentrating traffic on a small set of stable paths, ZooRoute resets these rankings every *30mins*.

ZooRoute is effective for most failures, but not a universal remedy. ZooRoute targets fast failure response, yet some anomalies remain outside its scope and require coordination with other systems: *i) Capacity scarcity and routing instability.* ZooRoute may underperform when the network has little spare capacity or when routing states are unstable, as shown in Case 3. In such situations, previously healthy candidates can quickly become invalid, and rerouting traffic may further increase congestion. In practice, ZooRoute reacts within seconds, serving as an emergency mitigation to reduce immediate tenant impact. Restoring normal service typically requires Traffic Engineering to recompute paths and rebalance load at a coarser timescale. The two are complementary and do not conflict. *ii) Low-rate probabilistic loss.* When packet loss is rare and random, a small number of probes

may not reveal the impairment reliably. Detecting such issues requires more probes or a longer observation window, introducing a trade-off between detection latency and telemetry overhead. *iii) Endpoint-adjacent faults.* ZooRoute is less effective when the packet loss originates near endpoints, since underlay path switching cannot eliminate the bottleneck. A representative example is microburst-induced drops at the RX side of a server NIC. In these cases, ZooRoute may change source ports frequently without reducing packet loss. While this phenomenon occurs daily, it affects only a negligible fraction of VTEPs. Operationally, we treat excessive switching rates as alarm signals and leverage automated Standard Operating Procedures (SOPs) alongisde engineers to diagnose and fix the endpoint issues. *iv) Failures inside the VM datapath.* ZooRoute operates at CSP-owned VTEPs and steers traffic by modifying the outer encapsulation. It does not directly cover faults within the VM's own datapath (*e.g.*, vNIC drops, queue hangs). Such failures require VM-granularity visibility and are better addressed by our proactive virtual network telemetry system Zoonet [81]. In production, ZooRoute and Zoonet are complementary: ZooRoute minimizes disruption by fast bypassing faulty paths, while Zoonet supports detection, diagnosis, tenant impact assessment, and guides repairs and architectural evolution.

Beyond fast recovery: latency optimization. Beyond failure recovery, ZooRoute can use the RTTs collected by its probes to select lower-latency paths under normal operation. Our online measurements show that the P99 latencies among paths between server pairs is 10-20μs within the same AZ, 100-150μs across AZs, and about 20ms across regions (Fig. A3). Since the first two cases are acceptable for most tenants, ZooRoute focuses on optimizes inter-region traffic. Specifically, it preferentially selects the 10% of available source ports with the lowest RTT and replaces them if their latency continues to increase for more than *2min*. Fig. A4 shows the latency improvement in the inter-region setting. Going forward, we will refine source port allocation by combining physical link utilization metrics, and address clock-synchronization limitations to improve latency estimation using one-way telemetry.

9 Related Work

Traditional approaches. Traditionally, network failures are handled in reactive ways, where new paths are recomputed at distributed routers based on routing policies [18, 66], or at an centralized controller via traffic engineering [34, 35, 64, 73, 80]. These processes are lengthy due to route propagation or interactions between the data plane and the controller. As an improvement, fast reroute [12, 27, 33, 52, 57] relies solely on the data plane to redirect traffic away from failure-affected locations. However, the precomputed backup paths are difficult to cover various kinds of failures in large-scale networks.

Failure recovery at underlay. Some RDMA recovery mechanisms modify the source port to steer UDP traffic onto alternative paths [5, 49, 65]. This breaks TCP connections and thus cannot extend to TCP traffic. Meanwhile, directly modifying original traffic is unsuitable in multi-tenant public clouds. Google's protective reroute [69] reroutes by modifying the IPv6 Flow Label, requiring a recent kernel version, an IPv6 underlay, and switch support for Flow Label hashing. Google notes that it currently works only for TCP and required substantial underlay upgrades [69]. Moreover, its trial-and-error nature poses risk to tenant SLAs. Programmable ECMP [46] can provide controllable hashing but is tightly coupled with device-specific capabilities. Some advanced traffic engineering approaches leverage divide-and-conquer to isolate failure domains and speed up route convergence [21, 34, 40–42], but they still require changes to the network architecture. Likewise, fine-grained load-balancing mechanisms such as ConWeave [62] and PLB [58] also reroute traffic to enhance network quality, yet they suffer from similar constraints—rely on specialized devices (*i.e.*, programmable switches or IPv6 underlay) and bound to specific transport protocols. In contrast, ZooRoute avoids these pitfalls and achieves device-independent recovery for all traffic types.

Failure recovery at overlay. CSPs develop systems like VNET Pingmesh [60], VTrace [24], and Zoonet [81] to monitor virtual networks and diagnose packet loss. However, they still rely on OCEs for tenant-aware failure repair. Resilient overlay routing approaches [11, 63, 72] reroute by steering traffic through overlay anchors to bypass failures. Technically, they are not mutually exclusive with ZooRoute: while ZooRoute modifies the outer source port to exploit underlay path diversity, they reroute by selecting different tunnel endpoints by modifying the outer destination IP. Clove [37] utilizes a hypervisor-based load balancing mechanism with path probing and source port altering, which we find is also viable to enhance network reliability. However, Clove was only evaluated in a small-scale testbed with 32 servers. Besides, its Clove-ECN and Clove-INT modes rely on device-specific features, hindering production deployment. In contrast, ZooRoute builds on standard ECMP and tackles resource challenges in diverse cloud network scenarios.

10 Conclusion

We present ZooRoute, Alibaba's fast failure recovery service that achieves end-to-end failure bypass within seconds in large-scale cloud networks. ZooRoute performs candidate path provisioning and proactive rerouting at the overlay layer that are transparent to tenants and agnostic to underlay devices. A range of optimization techniques improve its efficiency in path probing, recording, and switching to address the challenges at public cloud scale. Deployed in production for 26 months, ZooRoute has successfully handled various network failures without tenant awareness.

Acknowledgements. The authors would like to thank the shepherd Alex C. Snoeren and the anonymous reviewers for their con-structive comments.

Ethics. *This work does not raise any ethical issues.*

References

[1] Aeron - the global technology standard for high-throughput, low-latency, fault-tolerant trading systems. https://aeron.io/.

[2] Command line tool and library for transferring data with urls. https://curl.se/.

[3] Elastic compute service. https://www.alibabacloud.com/en/product/ecs.

[4] Inc gartner. invest implications: Cloud shift - 2023 through 2027. https://www.gartner.com/en/documents/4880831.

[5] Rdma over converged ethernet (roce). https://docs.nvidia.com/networking/display/ofed510660/rdma+over+converged+ethernet+(roce).

[6] Route tables. https://www.alibabacloud.com/help/en/vpc/vpc-route-table.

[7] Server load balancer. https://www.alibabacloud.com/en/product/server-load-balancer.

[8] Vpcs and vswitches. https://www.alibabacloud.com/help/en/vpc/vpc-and-vswitch.

[9] Sugam Agarwal, Murali Kodialam, and TV Lakshman. Traffic engineering in software defined networks. In *2013 Proceedings IEEE INFOCOM*, pages 2211–2219. IEEE, 2013.

[10] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 38(4):63–74, 2008.

[11] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 131–145, 2001.

[12] Alia Atlas and Alex Zinin. Basic specification for ip fast reroute: Loop-free alternates. Technical report, 2008.

[13] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring multipath routing in the internet. *IEEE/ACM Transactions on Networking*, 19(3):830–840, 2010.

[14] Deepak Bansal, Gerald DeGrace, Rishabh Tewari, Michal Zygmunt, James Grantham, Silvano Gai, Mario Baldi, Krishna Doddapaneni, Arun Selvarajan, Arunkumar Arumugam, et al. Disaggregating stateful network functions. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1469–1487, 2023.

[15] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[16] Yasmina Bouizem, Nikos Parlavantzas, Djawida Dib, and Christine Morin. Active-standby for high-availability in faas. In *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*, pages 31–36, 2020.

[17] Yanpei Chen, Rean Griffith, Junda Liu, Randy H Katz, and Anthony D Joseph. Understanding tcp incast throughput collapse in datacenter networks. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 73–82, 2009.

[18] Pei-chun Cheng, Xin Zhao, Beichuan Zhang, and Lixia Zhang. Longitudinal study of bgp monitor session failures. *ACM SIGCOMM Computer Communication Review*, 40(2):34–42, 2010.

[19] Michael Dalton, David Schultz, Jacob Adriaens, Ahsan Arefin, Anshuman Gupta, Brian Fahs, Dima Rubinstein, Enrique Cauich Zermeno, Erik Rubow, James Alexander Docauer, et al. Andromeda: Performance, isolation, and velocity at scale in cloud network virtualization. In *15th USENIX symposium on networked systems design and implementation (NSDI 18)*, pages 373–387, 2018.

[20] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *Proceedings of the 13th international conference on emerging networking experiments and technologies*, pages 160–166, 2017.

[21] Marek Denis, Yuanjun Yao, Ashley Hatch, Qin Zhang, Chiun Lin Lim, Shuqiang Zhang, Kyle Sugrue, Henry Kwok, Mikel Jimenez Fernandez, Petr Lapukhov, et al. Ebb: Reliable and evolvable express backbone network in meta. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 346–359, 2023.

[22] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. Maglev: A fast and reliable software network load balancer. In *Nsdi*, volume 16, pages 523–535, 2016.

[23] U. Elzur F. Maino, L. Kreeger. Generic protocol extension for vxlan (vxlan-gpe). https://www.ietf.org/archive/id/draft-ietf-nvo3-vxlan-gpe-12.html.

[24] Chongrong Fang, Haoyu Liu, Mao Miao, Jie Ye, Lei Wang, Wansheng Zhang, Daxiang Kang, Biao Lyv, Peng Cheng, and Jiming 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*, pages 31–43, 2020.

[25] Daniel Firestone. {VFP}: A virtual switch platform for host {SDN} in the public cloud. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 315–328, 2017.

[26] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, et al. Azure accelerated networking:{SmartNICs} in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, 2018.

[27] Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. Casa: congestion and stretch aware static fast rerouting. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 469–477. IEEE, 2019.

[28] Pankaj Garg and Y Wang. Nvgre: Network virtualization using generic routing encapsulation. Technical report, 2015.

[29] Supriyo Ghosh, Manish Shetty, Chetan Bansal, and Suman Nath. How to fight production incidents? an empirical study on a large-scale cloud service. In *Proceedings of the 13th Symposium on Cloud Computing*, pages 126–141, 2022.

[30] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference*, pages 350–361, 2011.

[31] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 139–152, 2015.

[32] Shixian Guo, Kefei Liu, Yulin Lai, Yangyang Bai, Ziwei Zhao, Songlin Liu, Jianghang Ning, Gen Li, Jianwei Hu, Yongbin Dong, et al. Bytetracker: An agentless and real-time path-aware network probing system. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 541–553, 2025.

[33] Thomas Holterbach, Stefano Vissicchio, Alberto Dainotti, and Laurent Vanbever. Swift: Predictive fast reroute. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 460–473, 2017.

[34] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, et al. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 74–87, 2018.

[35] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[36] Gueyoung Jung, Parisa Rahimzadeh, Zhang Liu, Sangtae Ha, Kaustubh Joshi, and Matti Hiltunen. Virtual redundancy for active-standby cloud applications. In *IEEE INFOCOM 2018-IEEE conference on computer communications*, pages 1916–1924. IEEE, 2018.

[37] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. Clove: Congestion-aware load balancing at the virtual edge. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 323–335, 2017.

[38] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*, pages 1–12, 2016.

[39] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan Jackson, et al. Network virtualization in multi-tenant datacenters. In *11th USENIX symposium on networked systems design and implementation (NSDI 14)*, pages 203–216, 2014.

[40] Alexander Krentsel, Nitika Saran, Bikash Koley, Subhasree Mandal, Ashok Narayanan, Sylvia Ratnasamy, Ali Al-Shabibi, Anees Shaikh, Rob Shakir, Ankit Singla, et al. A decentralized sdn architecture for the wan. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 938–953, 2024.

[41] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized cloud wide-area network traffic engineering with {BLASTSHIELD}. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 325–338, 2022.

[42] Umesh Krishnaswamy, Rachee Singh, Paul Mattes, Paul-Andre C Bissonnette, Nikolaj Bjørner, Zahira Nasrin, Sonal Kothari, Prabhakar Reddy, John Abeln, Srikanth Kandula, et al. {OneWAN} is better than two: Unifying a split {WAN} architecture. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 515–529, 2023.

[43] Jie Li, Vasileios Giotsas, Yangyang Wang, and Shi Zhou. Bgp-multipath routing in the internet. *IEEE Transactions on Network and Service Management*, 19(3):2812–2826, 2022.

[44] Xing Li, Xiaochong Jiang, Ye Yang, Lilong Chen, Yi Wang, Chao Wang, Chao Xu, Yilong Lv, Bowen Yang, Taotao Wu, et al. Triton: A flexible hardware offloading architecture for accelerating apsara vswitch in alibaba cloud. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 750–763, 2024.

[45] Xing Li, Enge Song, Bowen Yang, Tian Pan, Ye Yang, Qiang Fu, Yang Song, Yilong Lv, Zikang Chen, Jianyuan Lu, et al. Nezha: Smartnic-based virtual switch load sharing. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 218–233, 2025.

[46] Yadong Liu, Yunming Xiao, Xuan Zhang, Weizhen Dang, Huihui Liu, Xiang Li, Zekun He, Jilong Wang, Aleksandar Kuzmanovic, Ang Chen, et al. Unlocking {ECMP} programmability for precise traffic control. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 87–106, 2025.

[47] Jianyuan Lu, Tian Pan, Shan He, Mao Miao, Guangzhe Zhou, Yining Qi, Shize Zhang, Enge Song, Xiaoqing Sun, Huaiyi Zhao, et al. Cloudsentry: Two-stage heavy hitter detection for cloud-scale gateway overload protection. *IEEE Transactions on Parallel and Distributed Systems*, 35(4):616–633, 2023.

[48] Jianyuan Lu, Shunmin Zhu, Jun Liang, Yuxiang Lin, Tian Pan, Yisong Qiao, Yang Song, Wenqiang Su, Yixin Xie, Yanqiang Li, et al. Albatross: A containerized cloud gateway platform with fpga-accelerated packet-level load balancing. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 71–84, 2025.

[49] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. {Multi-Path} transport for {RDMA} in datacenters. In *15th USENIX symposium on networked systems design and implementation (NSDI 18)*, pages 357–371, 2018.

[50] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T Sridhar, Mike Bursell, and Chris Wright. Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks. Technical report, 2014.

[51] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 15–28, 2017.

[52] Ping Pan, George Swallow, and Alia Atlas. Fast reroute extensions to rsvp-te for lsp tunnels. Technical report, 2005.

[53] Tian Pan, Kun Liu, Xionglie Wei, Yisong Qiao, Jun Hu, Zhiguo Li, Jun Liang, Tiesheng Cheng, Wenqiang Su, Jie Lu, et al. {LuoShen}: A {Hyper-Converged} programmable gateway for {Multi-Tenant}{Multi-Service} edge clouds. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 877–892, 2024.

[54] Tian Pan, Enge Song, Yueshang Zuo, Shaokai Zhang, Yang Song, Jiangu Zhao, Wengang Hou, Jianyuan Lu, Xiaoqing Sun, Shize Zhang, et al. Hermes: Enhancing layer-7 cloud load balancers with userspace-directed i/o event notification. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 363–380, 2025.

[55] Tian Pan, Nianbing Yu, Chenhao Jia, Jianwen Pi, Liang Xu, Yisong Qiao, Zhiguo Li, Kun Liu, Jie Lu, Jianyuan Lu, et al. Sailfish: Accelerating cloud-scale multi-tenant multi-service gateways with programmable switches. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 194–206, 2021.

[56] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open {vSwitch}. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, pages 117–130, 2015.

[57] Kun Qiu, Jin Zhao, Xin Wang, Xiaoming Fu, and Stefano Secci. Efficient recovery path computation for fast reroute in large-scale software-defined networks. *IEEE Journal on Selected Areas in Communications*, 37(8):1755–1768, 2019.

[58] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. Plb: congestion signals are simple and effective for network load balancing. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 207–218, 2022.

[59] Matthew Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 248–258, 2003.

[60] Arjun Roy, Deepak Bansal, David Brumley, Harish Kumar Chandrappa, Parag Sharma, Rishabh Tewari, Behnaz Arzani, and Alex C Snoeren. Cloud datacenter sdn monitoring: Experiences and challenges. In *Proceedings of the Internet Measurement Conference 2018*, pages 464–470, 2018.

[61] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. A cloud-optimized transport protocol for elastic and scalable hpc. *IEEE micro*, 40(6):67–73, 2020.

[62] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. Network load balancing with in-network reordering support for rdma. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 816–831, 2023.

[63] David Sontag, Yang Zhang, Amar Phanishayee, David G Andersen, and David Karger. Scaling all-pairs overlay routing. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 145–156, 2009.

[64] Sucha Supittayapornpong, Barath Raghavan, and Ramesh Govindan. Towards highly available clos-based wan routers. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 424–440. 2019.

[65] Feng Tian, Yang Zhang, Wei Ye, Cheng Jin, Ziyan Wu, and Zhi-Li Zhang. Accelerating distributed deep learning using multi-path rdma in data center networks. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pages 88–100, 2021.

[66] Lan Wang, Malleswari Saranu, Joel M Gottlieb, and Dan Pei. Understanding bgp session failures in a large isp. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 348–356. IEEE, 2007.

[67] Zilong Wang, Layong Luo, Qingsong Ning, Chaoliang Zeng, Wenxue Li, Xinchen Wan, Peng Xie, Tao Feng, Ke Cheng, Xiongfei Geng, et al. {SRNIC}: A scalable architecture for {RDMA}{NICs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1–14, 2023.

[68] Chengkun Wei, Xing Li, Ye Yang, Xiaochong Jiang, Tianyu Xu, Bowen Yang, Taotao Wu, Chao Xu, Yilong Lv, Haifeng Gao, et al. Achelous: Enabling programmability, elasticity, and reliability in hyperscale cloud networks. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 769–782, 2023.

[69] David Wetherall, Abdul Kabbani, Van Jacobson, Jim Winget, Yuchung Cheng, Charles B Morrey III, Uma Moravapalle, Phillipa Gill, Steven Knight, and Amin Vahdat. Improving network availability with protective reroute. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 684–695, 2023.

[70] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath {TCP}. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[71] Shinae Woo, Justine Sherry, Sangjin Han, Sue Moon, Sylvia Ratnasamy, and Scott Shenker. Elastic scaling of stateful network functions. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 299–312, 2018.

[72] Bingyang Wu, Kun Qian, Bo Li, Yunfei Ma, Qi Zhang, Zhigang Jiang, Jiayu Zhao, Dennis Cai, Ennan Zhai, Xuanzhe Liu, et al. Xron: A hybrid elastic cloud overlay network for video conferencing at planetary scale. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 696–709, 2023.

[73] Dingming Wu, Yiting Xia, Xiaoye Steven Sun, Xin Sunny Huang, Simbarashe Dzinamarira, and TS Eugene Ng. Masking failures from application performance in data center networks with shareable backup. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 176–190, 2018.

[74] Xipeng Xiao, Alan Hannan, Brook Bailey, and Lionel M Ni. Traffic engineering with mpls in the internet. *IEEE network*, 14(2):28–33, 2000.

[75] Tingting Xu, Bengbeng Xue, Yang Song, Xiaomin Wu, Xiaoxin Peng, Yilong Lyu, Xiaoliang Wang, Chen Tian, Baoliu Ye, Camtu Nguyen, et al. {CyberStar}: Simple, elastic and {Cost-Effective} network functions management in cloud network at scale. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 227–246, 2024.

[76] Yunhong Xu, Keqiang He, Rui Wang, Minlan Yu, Nick Duffield, Hassan Wassel, Shidong Zhang, Leon Poutievski, Junlan Zhou, and Amin Vahdat. Hashing design in modern networks: Challenges and mitigation techniques. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 805–818, 2022.

[77] Wensong Zhang et al. Linux virtual server for scalable network services. In *Ottawa Linux Symposium*, volume 2000, 2000.

[78] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. Hashing linearity enables relative path control in data centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 855–862, 2021.

[79] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 418–432, 2021.

[80] Zhizhen Zhong, Manya Ghobadi, Alaa Khaddaj, Jonathan Leach, Yiting Xia, and Ying Zhang. Arrow: restoration-aware traffic engineering. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 560–579, 2021.

[81] Shunmin Zhu, Jianyuan Lu, Biao Lyu, Tian Pan, Chenhao Jia, Xin Cheng, Daxiang Kang, Yilong Lv, Fukun Yang, Xiaobo 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*, pages 321–336, 2022.

Appendices

A ZooRoute Protocol Specification

To ensure reliable operation across massive VTEPs in large CSPs, we design a protocol for ZooRoute agents guided by the following key principles.

Telemetry veracity. To accurately probe tenant traffic paths, ZooRoute constructs UDP-based probe packets aligned with the VXLAN encapsulation used in Alibaba Cloud. Besides, in production, device hot upgrades or live migrations may change the states of probing processes, compromising path analysis consistency and introducing noises unrelated to the actual network path status. ZooRoute embeds a probe process index in the packet body as a unique identifier to detect process changes and filter out the resulting dirty data.

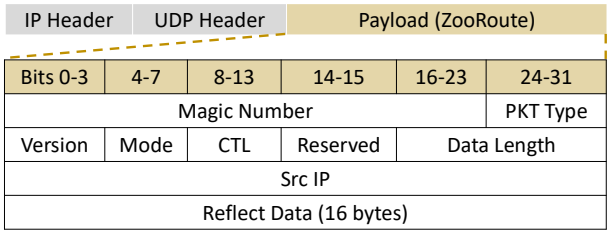
Partial deployment support. In production, new features roll out gradually across versions. Therefore, the source VTEP issues a negotiation request before probing to clarify its function version with the destination VTEP.

VTEP controllability. ZooRoute is a distributed system runs at VTEP side; each VTEP must observe and regulate its probing tasks. We add a "CTL" field with four modes: 1) default for normal probing; 2) RST to re-negotiate between source and destination; 3) BP for backpressure to throttle the source's probing frequency; 4) FIN to terminate the probing task.

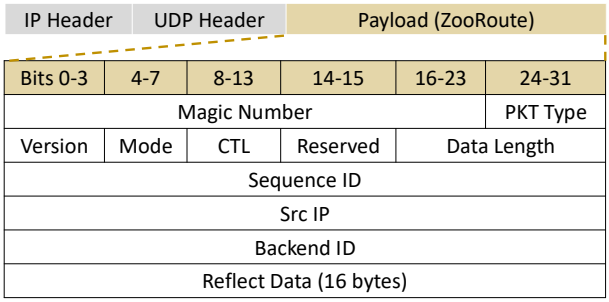
Heterogeneous devices compatibility. To simply telemetry packet processing, ZooRoute sets a "Reflect data" field at the tail of each packet. This field uniformly covers diverse information that only needs to be processed by source hosts. Different types of source hosts can define the contents of this field independently, for example, sport, process ID, *etc*. This design simplifies the format of ZooRoute protocol, making it easily compatible with heterogeneous underlying devices in large CSPs. Besides, it curtails the parsing burden on target hosts, which is important for hardware devices.

The detailed explanation of each field in ZooRoute protocol is as follows:

- **Magic Number**: set as *0xffeedd* to distinguish ZooRoute packets from those of other business traffic.
- **PKT Type**: indicates packet type with *0x11* for negotiation request, *0x18* for negotiation response, *0x7E* for telemetry request, *0x7F* for telemetry response, and *0x05* for path table updates. Note that, for VTEPs that do not support ZooRoute, the "Magic Number" field overlaps with the "Flags" field in VXLAN-GPE [23], and "Pkt Type" will be parsed as "Next Protocol". Therefore, values in "Next Protocol" field of VXLAN-GPE, like *0x1*, *0x02*, should not be picked as PKT Type.
- **Version**: identifies the version number of current protocol.
- **Mode**: for negotiation packets, this field indicates all telemetry modes supported by the VTEP. For telemetry packets, it shows the current mode.
- **CTL**: this field is utilized by VTEPs to control probing tasks, with *0x0* as the default value for normal probing; *0x1* as RST to let the source VTEP reconduct negotiation, *0x2* as BP, telling the source VTEP to reduce telemetry frequency; *0x4* as *FIN*, making the source VTEP stop probing.
- **Data Length**: total length of the data packet.
- **Sequence ID**: each packet is assigned a sequence number, starting from 0, to detect packet disorder and provide timely feedback when problems occur on the current probing path.
- **Src IP**: IP address of the source VTEP.
- **Backend ID**: as described in §6, this field records MAC address of the middlebox in stateful forwarding scenarios.
- **Reflect Data**: This field carries source-only metadata and is ignored by the destination VTEP. Different types of source VTEPs can define its own schema. For example, It may include source port for path recording. Besides, the source VTEP can record a probe-process ID, so that dual processes during hot upgrades or live migration can be disambiguated, preserving the accuracy of path probing.



(a) Negotiation packet (request & response)



(b) Telemetry packet (request & response)

Figure A1: Packet format of ZooRoute protocol.

B Additional Tables and Figures

Table A1: Root causes of failures with unusually long detection time (*d_time*) or recovery time (*r_time*) in Alibaba Cloud from 2022-01-01 to 2023-03-3.

Root causes	Avg. d_time	Avg. r_time	# of cases
ISP network failure	22.4	99	13
Physical switch card exception	38.8	65.7	6
Physical switch CPU core overload	17.5	304	2
Network change or upgrade	31	16.5	2
Configuration error or inconsistency	13	27	1
In total			**24**

Table A2: Packet retransmission and CPU utilization under different degrees of packet out-of-order.

Traffic	Path switching	# of retransmission	CPU usagesaa
2Gbps	No changes	0	28.0%
	N = 64	0	29.4%
	N = 32	0	32.1%
	N = 16	0	35.1%
	N = 1	0	41.2%
4Gbps	No changes	0	49.9%
	N = 64	0	51.5%
	N = 32	0	55.7%
	N = 16	0	61.2%
	N = 1	0	70.0%

Table A2 explanation. While path switching bypasses failures, it may introduce packet reorder. We therefore stress-test whether such out-of-order delivery harms transport. Using Alibaba Cloud's most common VM instance (ecs.g7.large), we generate 2Gbps and 4Gbps TCP traffic and synthetically induce out-of-order by switching the path every $N \in \{64, 32, 16, 1\}$ packets. Results indicate that Linux kernel's in-order reassembly tolerates the induced out-of-order without triggering retransmissions. When dealing with mild out-of-order (N=64), which is already more severe than ZooRoute would cause, the CPU usage only increases by less than 2% relative to the no-switch baseline at both rates. Techniques mitigating severe out-of-order in load balancing or multipath settings [61,62,67] are orthogonal to ZooRoute, they can complement each other to respectively ensure fast failure recovery and enhance network performance.

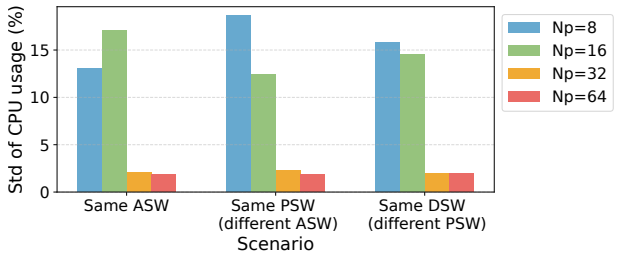


Figure A2: CPU load balance of destination VTEPs in various src-dst server pair placements under different N_p.

Fig. A2 explanation. ZooRoute prefers a more orthogonal source port selection to prevent traffic from concentrating on a single underlay path, which could render many source ports simultaneously unavailable if that path fails. Although hundreds of ECMP paths exist between servers, traffic to a single server is ultimately hashed by the NIC onto a limited number of CPU cores. As long as CPU utilization remains balanced, the likelihood of network-level traffic polarization is further reduced. Therefore, we set N_p based on the CPU usage distributions of the destination VTEPs. Alibaba Cloud employs a standard three-tier regional topology, with Top-of-Rack access switches (ASW), Points-of-Delivery aggregation switches (PSW), and core Distribution switches (DSW). In the experiment, we randomly select 10 src-dst server pairs for each placement scenarios: *i*) same ASW, *ii*) same PSW but different ASWs, and *iii*) same DSW but different PSWs. Fig. A2 reports, under identical traffic, the mean standard deviation (std) of CPU utilization at the destination while varying $N_p \in \{8, 16, 32, 64\}$. We find that $N_p = 32$ achieves near-optimal load balance across all scenarios. Increasing to $N_p = 64$ yields only marginal gains (<0.4% reduction in std) while roughly doubling telemetry overhead; hence we default to $N_p = 32$. Before and after ZooRoute deployment, no significant change in link utilization distribution is observed.

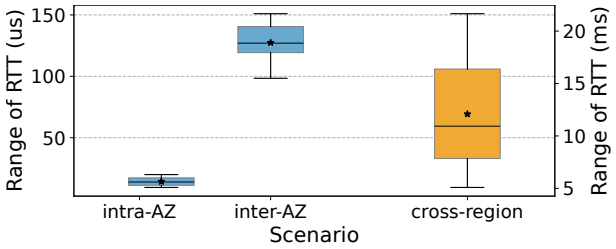


Figure A3: Distribution of RTT range (*i.e.*, $RTT_{max} - RTT_{min}$) in different scenarios.

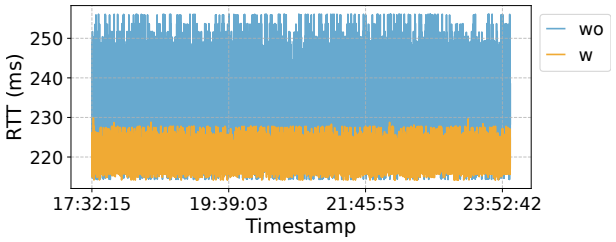


Figure A4: RTT between cross-region gateways w/wo ZooRoute guided path selection.

Fig. A3 explanation. We randomly select 50 pairs of VTEPs for intra-AZ, inter-AZ, and cross-region scenarios. For each pair, the source VTEP changes the source port 1,000 times to traverse diverse ECMP paths. We compute P999 RTT ranges (*i.e.*, $RTT_{max} - RTT_{min}$) per pair and plot the distributions. The RTT range is 10~20*us* intra-AZ and 100~150*us* inter-AZ, which is acceptable for such deployments. Cross-region pairs exhibit much larger ranges, up to 20*ms*, indicating substantial headroom for ZooRoute-guided "better path" selection.

Fig. A4 explanation. This figure shows RTT between gateways in two different regions under normal operations. We can see that ZooRoute-guided path selection reduces RTT by more than 20*ms* (about 10%) compared to the unguided baseline. Our current heuristic is simple: select the top 10% source ports with the lowest latency and drop a port if its telemetry shows persistent fluctuation for more than 2*min*. This is being A/B tested with a small subset of major tenants. We plan to incorporate underlying link-utilization distributions to further stabilize path selections, ensuring that traffic rerouting does not induce RTT oscillations.