USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Harp: Improving VPC Network Availability via Efficient Failure Detection and Rerouting in Tencent Cloud

Jiayu Hu, Feng Jin, and Xianping Zhou, *Tencent;* Kai Zhang, *Fudan University;* Zhen Shen and Yongkang Luo, *Tencent*

https://www.usenix.org/conference/nsdi26/presentation/hu-jiayu

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

Harp: Improving VPC Network Availability via Efficient Failure Detection and Rerouting in Tencent Cloud

Jiayu Hu[1], Feng Jin[1], Xianping Zhou[1], Kai Zhang[2†], Zhen Shen[1], Yongkang Luo[1]
[1]*Tencent,* [2]*Fudan University*

Abstract

We present Harp, a mechanism for efficient and robust failure detection and recovery for VPC networks. Unlike previous approaches, Harp does not rely on specific hardware features or transport protocols. Instead, Harp is a lightweight software implementation that identifies feasible physical paths between each pair of communicating hosts and monitors their health states. When network failures occur, Harp can swiftly switch influenced flows from a failed path to a healthy path within a short interval. Harp has been deployed in Tencent Cloud for more than two years and has proven its effectiveness in handling network failures. According to the results from production systems, Harp can significantly reduce the VPC network's outage time by 78.71%-99.97% and essentially bypass failed paths on a sub-second timescale. We also share our experiences of how Harp efficiently detects and handles network failures when deployed in Tencent Cloud.

1 Introduction

Tencent Cloud is a large cloud service provider that offers computational services for global clients, as well as Tencent's flagship products, e.g., WeChat and QQ. Tencent Cloud provides rich cloud services such as computing, storage, network, and databases. To meet the dramatically increasing demand for cloud computing, hundreds of thousands of servers are deployed in Tencent data centers worldwide. However, dozens of network failures inevitably occur at Tencent data centers every year due to hardware switch faults, software bugs, and operational configuration mistakes. These failures cause jitters in networks and even result in disconnection. Such failures would incur service interruptions of Tencent Cloud products and compromise service-level agreements (SLAs). Therefore, the mechanism to effectively detect and handle network failures has been a major focus at Tencent Cloud.

Because of the importance of network failure handling in delivering robust cloud services, there has been a spectrum of techniques deployed across main cloud providers, such as MPTCP [23, 39], SRD [43], and PRR [53]. These studies are generally based on multi-path technologies to leverage abundant physical paths between communicating hosts. Specifically, they utilize the forwarding policies at switches to route packets across multiple paths in the data center network. For instance, MultiPath TCP (MPTCP) creates multiple TCP sub-flows to map the traffic on separate physical paths, while Protective ReRoute (PRR) uses IPv6 flow labels to inform switches to route flows to new paths. Both approaches use TCP Retransmission Time-Out (RTO) as a signal of failure. When there is a failure, the approaches change their flow identifiers, such as the IPv6 flow label or a field in the five-tuple, to try to switch to a new path. These approaches have demonstrated their effectiveness when deployed in the corresponding clouds.

Tencent Cloud does not adopt the existing approaches in handling network failures in its data centers. The main reasons are as follows. 1) **Legacy network devices**: There are many legacy network devices deployed across Tencent data centers that do not support unified global configurations. For instance, not all switches support the IPv6 protocol. Therefore, techniques using IPv6 flow labels cannot be adopted here. 2) **Incompatible with existing software**: Transport-based approaches, such as MPTCP, require modification of the VM software stack. Due to incompatibility with existing cloud software, deploying them at scale is challenging. 3) **Unpredictable recovery time**: With the emergence of latency-sensitive applications, such as Redis, tenants demand higher efficiency for failure detection and recovery. Although existing approaches try to switch physical paths when they detect a network failure, it is still possible that the new path also suffers from the same network issue. Therefore, considering the underlying hardware infrastructure and the evolving latency requirements of cloud applications, Tencent Cloud aims to design a general, effective, yet lightweight failure detection and handling mechanism that fits the modern cloud environment.

This paper proposes Harp, a lightweight failure detection

†Corresponding author.

and recovery mechanism that effectively enhances the availability of VPC networks in Tencent Cloud. The key to Harp is to identify available physical paths between each pair of communicating hosts and balance the flows among these paths to enhance network utilization. Harp uses the UDP source ports to leverage Equal Cost Multi-Path (ECMP) [48, 60] on switches to control the physical paths, which helps build a path pool for each host pair (§3.2). Based on this approach, Harp adopts a lightweight in-band failure detection mechanism to monitor the health states of physical paths in real-time (§4). Specifically, Harp periodically inserts probe information into the traffic flow in both directions and uses a mechanism to calculate whether the path is functioning or has failed. When a network failure happens, Harp switches the affected flow's physical path to a healthy one by modifying its UDP source port (§3.3). By knowing the states of all physical paths, Harp can avoid unavailable paths and reroute affected flows to healthy ones with a significantly higher probability. By in-band detection, Harp can achieve high probe coverage with negligible overhead (§4.3). Harp does not rely on specific hardware features and is transparent to cloud applications, enabling the deployment of Harp on existing infrastructures.

Harp has been deployed in Tencent Cloud since 2023, across nearly all regions. We share our deployment experiences with three typical cases, including core switch and DCI switch failures (§5 and §6). Each case details how Harp efficiently handles network failures for cloud tenants. Furthermore, we have evaluated Harp's effectiveness in ultra-large production systems (§7). According to the results, Harp can significantly reduce the outage time for VPC networks by 78.71%-99.97% and essentially reroute flows to functional paths on a sub-second timescale.

2 Background

2.1 Tencent Data Center Network Architecture

Data Center Network in Tencent: Tencent Cloud services have been provided since 2013, and so far, hundreds of thousands of servers have been deployed in Tencent data centers worldwide. In Tencent data centers, a Data Center Network (DCN) is organized as a modified tree structure [2, 25], as shown in Fig. 1. The tree-based structure comprises three tiers of switches. Switches at the edge layer, known as leaf switches, are used to connect hosts with the network. Switches at the aggregation layer, known as spine switches, aggregate traffic from the edge layers and forward it to core switches that reside at the root of the tree. Core switches provide connectivity to spine switches and data-center interconnection (DCI) switches. DCI switches connect Tencent Cloud's data centers. Each host is connected to two leaf switches via network interface cards (NICs) that are aggregated into a single logically bonded interface [31, 37]. The two-leaf switches aggregate host traffic in a rack and forward it upstream to the spine switches. The two tiers of leaf and spine switches, and the resident hosts, compose a point of delivery (PoD).

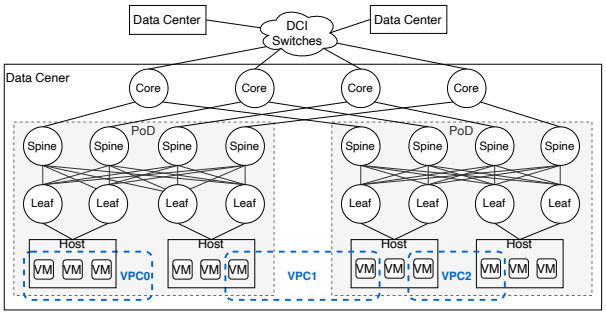

Figure 1: A data center network topology in Tencent Cloud

Virtual Private Cloud: Virtual Private Cloud (VPC) [3, 54, 57] is one of the fundamental network techniques of infrastructure-as-a-service (IaaS) provided by cloud service providers, like Tencent, Amazon Web Services, and Microsoft Azure. It provides an abstraction of allocated cloud resources and enables tenants to customize network space in the public cloud as a private network in enterprises. Lots of cloud services, like Redis [6], Cloud Object Storage (COS) [55], and Cloud File Storage (CFS) [27], have already been deployed with VPC services. VPC networks are unrelated to physical network typologies. For example, in Fig. 1, three VPCs connect Virtual Machines (VMs) deployed on the same host (VPC0), across hosts (VPC2), and across PoDs (VPC1). There is a class of tunnel protocols, which support the delivery of VM packets with the host IP address, used to implement virtual networks, such as Virtual Extensible LAN (VXLAN) [33]. In Tencent Cloud, the VPC service is built upon Generic Protocol Extension for VXLAN (VXLAN-GPE) [20]. VXLAN-GPE encapsulates VM traffic within UDP datagrams, and it is comprised of a UDP header, a VXLAN-GPE header, and multiple network service headers (NSHs) [38], including the Harp protocol header.

The VPC architecture in Tencent Cloud is depicted in Fig. 2. Every host is installed with three modules: a VPC agent, a virtual software-based switch, and our failure recovery module, Harp. The VPC agent is a distributed control plane module in charge of obtaining routing and access control configurations from a centralized controller of Software Defined Network (SDN) [24]. The virtual switch [35] works in the data plane, switching traffic among VMs and aggregating the external access traffic to leaf switches via querying routing tables maintained by the VPC agent. A VPC is identified and isolated via a unique VPC ID. When communicating inside a VPC, the virtual switch sends the egress VM traffic over VXLAN-GPE, where the VPC ID is stored in the NSH header. At the other end, the virtual switch checks whether ingress packets are destined for VMs in the host by querying mappings of VPC IDs and VPC subnets. It removes the outer headers before delivering the packets to destination VMs. In Tencent Cloud,

tenants can use either IPv4 or IPv6 as the inner IP protocol, but the underlying IP protocol is fixed to IPv4. This is due to commercial reasons, where parts of the network devices in Tencent data centers don't support IPv6.

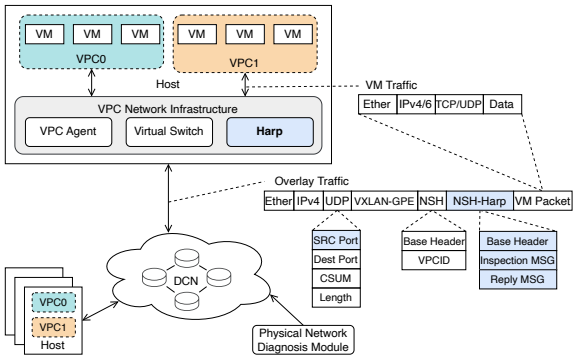

Figure 2: The VPC architecture in Tencent Cloud

2.2 Network Failures in Tencent Cloud

Network Failure Classification: There are various network failures in a production system that may have a significant impact on the SLAs of cloud services. Fig. 3 depicts the distribution of network failures that occurred in Tencent data centers in a year, which can be categorized into five classes. The errors on leaf switches, spine switches, core switches, and DCI switches accounted for 94% of all failures. Switch failures are usually caused by hardware issues, such as damaged optical modules and boards, interface down, unplanned reboots [15], link flapping [16, 36], and silent packet drops [4, 17]. Moreover, firmware and system software bugs [15] in switches also cause switch failures. Another type of event that results in network failures is configuration errors, such as incorrect Access Control List (ACL) rules [56] and black holes [17] due to routing misconfigurations. These account for 6% of the overall failures.

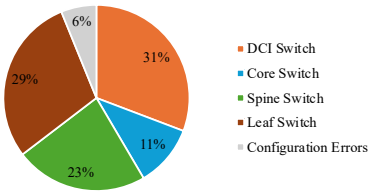

Figure 3: Network failure distribution in Tencent Cloud

Physical network failure diagnosis in Tencent Cloud: The network operational system in Tencent data centers has a built-in mechanism to detect network failures. It characterizes network status by monitoring key metrics for quality, like end-to-end round-trip delays, packet loss rate, and host availability. To get the network status in real-time, the system periodically pings servers to obtain the network latency and packet drop rate. Furthermore, it also enables switches to report abnormal events proactively. Once abnormal events are detected, it reports warnings and triggers the diagnosis process. In the diagnosis process, the network operational system analyzes the fault events and collects traces. After identifying fault elements, it moves to the isolation procedure, which migrates existing flows away from the faulty network elements. By adopting new paths that bypass the faulty elements, the network traffic of impacted tenants can recover from the failure. Finally, networking specialists are involved in analyzing root causes offline. The average time taken from discovering failure to recovering the business traffic is tens of seconds. Some faults, such as gray failures [19, 47], are difficult for the system to identify but are perceived and reported by our clients. In this situation, locating the faulty elements may take longer, which would increase the outage time.

2.3 Motivation

Although the physical network failure diagnosis mechanism has been functioning properly in Tencent Cloud for a few years, it cannot address the new client requirements. Specifically, with the emergence of latency-sensitive applications, tenants demand higher efficiency for failure detection and recovery. For example, the Tencent Redis service promises five 9s SLA [9], and it considers Redis operations as failed if replies are not returned within 200 ms. Tencent CFS service promises sub-millisecond latency [7]. The SLA of the Tencent COS service is 99.995% [8], and a network failure that lasts more than three minutes will exceed the monthly downtime allowed. The performance requirements of latency-sensitive applications demand that the VPC network's connectivity can be restored on a sub-second timescale. Two main classes of studies tackle network failures, i.e., the physical network diagnosis approach and the multi-path approach. However, we find that neither of them fits Tencent Cloud.

Physical network diagnosis approaches have an unacceptably high latency. Many studies [4, 11, 17, 18, 28, 34, 42, 47, 50, 56, 58, 63] focus on physical network failure diagnosis. Those approaches detect and localize faulty network components by collecting system metrics (e.g., latency, switch/port counters, and TCP metrics) and analyzing the data via a centralized agent. For example, Pingmesh [17] and NetNORAD [11] analyze network latency and packet drop rate collected by the ping approach. After localizing failures, NetPilot [56] and SWARM [34] further restart or deactivate the offending components to mitigate the impacts of failures. However, those approaches take tens of seconds or even minutes to locate failures. Like the approaches used in Tencent Cloud, localizing failures is time-consuming and cannot meet the latency requirements of latency-sensitive cloud applications. Other hardware-based approaches [5, 13, 21, 26, 29, 51] require support from special hardware. Considering the huge capital expenses and time efforts taken in the deployment, they are difficult to adopt in an ultra-large data center network like Tencent Cloud.

Existing multi-path approaches do not fit Tencent in-

frastructure. There is a spectrum of studies on failure detection and recovery in data centers via rerouting, such as MPTCP, SRD, and PRR. These studies leverage abundant physical paths between communicating end hosts to quickly recover from failures, while generally using different techniques. For instance, MPTCP creates multiple sub-flows for every TCP connection and migrates traffic from faulty paths into sub-flows on healthy paths in the event of network failures. PRR updates the Flow Label field in IPv6 headers to inform underlay switches to switch packets into new paths. Because of the swift path switching, multi-path approaches generally have millisecond-level latency, which is much lower than that of physical network diagnosis approaches. However, they cannot be directly adopted in Tencent Cloud for three reasons. First, many legacy network devices deployed across Tencent data centers don't support unified configuration due to commercial reasons. For example, not all switches support the IPv6 protocol, so techniques based on the IPv6 flow label cannot be used. Second, transport-based approaches require modification to the VM software stack, which can't be adopted transparently. For instance, enabling PRR in cloud environments requires the guest OS to support PRR. Thus, deploying them at scale is challenging. Third, the time to recover from failures is unpredictable. As errors may occur on the overlapped links between the faulty and newly selected paths, it is still possible that the packets suffer from the same network issues after re-path.

Therefore, there is an urgent need to develop an effective failure-handling approach in Tencent Cloud to achieve fast recovery after failures and meet increasingly stringent SLA requirements.

3 Design of Harp

We propose Harp, a lightweight failure detection and recovery mechanism for VPC networks in Tencent Cloud. In this section, we overview Harp's working mechanism and discuss its key design schemes, including path control, flow mapping, and failure handling.

3.1 Overview

The approach of Harp: The key idea of Harp is to initiate deterministic physical paths between each pair of communicating hosts, monitor the health of the paths in real-time, and swiftly switch to healthy paths when there is a failure. Since Tencent data centers adopt a tree-like topology, there are multiple available physical paths between each pair of hosts. Harp controls the paths for a pair of hosts by setting the UDP source port of the flows to get different switch-forwarding results. With this approach, Harp creates a path pool between each pair of hosts. Harp monitors the health states of paths in real-time by injecting probe information into them and detecting responses. Therefore, it knows which paths suffer failures and even the congestion level by calculating the latency of responses. When a network failure occurs, Harp sets another healthy path in the path pool to restore the network for affected flows.

Harp has three advantages. First, it monitors the health of physical paths, enabling it to reroute flows to healthy paths quickly. Second, it does not require special hardware support, so it can be directly deployed on existing infrastructure. Third, it is transparent to cloud applications. As a result, it can be applied to traffic based on any protocol. These features are critical to meeting the stringent and specific management and performance requirements in Tencent Cloud.

The workflow of Harp: Fig. 4 shows the workflow of Harp. Harp is implemented in the VPC network infrastructure layer of the end host. Harp has four main modules: the Path Control Module, the Flow Mapping Module, the Failure Detection Module, and the Failure Handling Module. During initialization, the Path Control Module creates a Path Pool for a pair of communicating hosts. The pool consists of the configuration for all available physical paths to the destination host. The Flow Mapping Module aims to distribute flows evenly across available paths in the Path Pool. The Flow Mapping Module selects a physical path for each packet/flow and transmits it through the network by encapsulating it with the corresponding UDP source port of the selected path. The Failure Detection Module runs to monitor and detect faulty paths in real-time. The Failure Detection Module in the source host inserts inspection packets into the paths to the destination host. The Failure Detection Module in the destination host records information for all paths and regularly sends replies to notify the source host. With the reply information, the Failure Detection Module in the source host examines whether all used paths are functional and marks the failed paths. When a path fails, the Failure Handling Module updates the path-mapping configuration to switch to a new healthy path. The Failure Detection Module also detects path restoration to migrate traffic back to repaired paths.

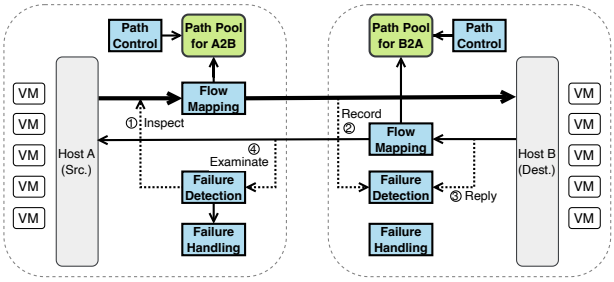

Figure 4: The workflow of Harp

The scope of Harp: Harp collaborates with the physical network diagnosis module to accelerate failure recovery for VPC networks. Working swiftly, Harp detects unavailable paths and mitigates failures by rerouting the affected VM traffic to healthy paths. After the network operational team repairs the offending component using physical network diag-

nosis approaches (§2.2), Harp migrates VM traffic back to the repaired paths. Harp targets network failures related to packet loss. Harp covers network device and link failures [15], and NIC errors on end hosts. It does not cover packet corruption failures, such as frame checksum errors.

3.2 Deterministic Path Control

Physical paths between end hosts: In a Tencent Cloud data center, the end-to-end path is typically composed of three or five hops, while a path involves more hops when it crosses data centers. Inside a PoD, there are three hops between two hosts, such as the green line, consisting of two leaf nodes and one spine node. In Fig. 5, with four paths between each pair of leaf nodes and two leaf nodes for each host, there are a total of 16 physical paths from Host1 to Host2. For hosts crossing PoDs, the blue line in Fig. 5 shows a path consisting of five hops between Host1 and Host4, and there are also a total of 16 physical paths. When a failure occurs in the network, it may affect one or a few paths, while the rest remain available and ready to be used. For instance, if a spine switch fails, four paths between Host1 and Host4 are affected, while eight paths stop functioning when a leaf switch is down.

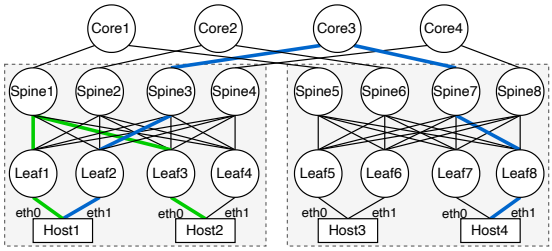

Figure 5: Multiple physical paths between hosts

Deterministic port-based path control: Harp leverages UDP source ports to map flows to a physical path. Harp identifies all physical paths between hosts and their represented port numbers to build a path pool. By monitoring the path state, Harp distributes flows among the healthy paths to enhance network resource utilization. Unlike previous approaches, Harp features fast path switching among distinct physical paths in the event of network failures. When a path fails, Harp picks a new port number, which always represents a different physical path to recover the traffic. Instead, without deterministic path control, different ports may lead to the same physical path, which would result in a less efficient failure recovery process.

In Tencent data centers, switches use ECMP as the routing scheme, and setting a field in a packet's five-tuple would influence the switches' forwarding decisions. Based on the linearity of ECMP hashing algorithms on switches proposed in [59], Harp leverages UDP source ports to control physical paths for VM traffic. Its primary mechanism is described as follows. Denoting S_i as the switch at hop i, the next-hop S_{i+1} is decided by the ECMP selection at S_i. Given hop i with N_i equal-cost next hops, the ECMP routing algorithm takes a 5-tuple of packet headers as input, and the routing results range from 0 to $N_i - 1$. For overlay traffic between a host pair in our data centers, the 5-tuple of an underlay header is fixed except for the UDP source port, so Harp utilizes UDP source ports to direct flows through paths. We divide the UDP source port (denoted as P_s) into N_i groups (denoted as SG_i) based on the results of ECMP by Equation 1, where hdr_0 is the random packet header seed, and the ECMP hash algorithm is XOR or CRC. G_k^i denotes a UDP source port group that generates the ECMP result k at hop i, where k is from one to the number of next-hop switches.

$$\begin{aligned} SG_i &:= \{\, G_k^i \mid 0 \le k \le N_i - 1 \,\} \\ G_k^i &:= \{\, P_s \mid ECMP(P_s \oplus hdr_0) = k \,\} \end{aligned} \tag{1}$$

By validating on switches in our data centers, applying the UDP source port in G_k^i can route a flow to the same next-hop, while source ports in different groups route the flow to different next-hops. Furthermore, considering paths with M hops, denoted as $Src \to S_1 \to S_2 \to \cdots \to S_M \to Dest$, Harp calculates the intersections ($\cap$) of G_k^i for switches from hop 1 to hop $M-1$ by Equation 2. We denote the resulting UDP source port collections as a set of *rPath*s. For VM traffic between one host pair, the UDP source ports in one *rPath* can always direct a flow to the same physical path, while ports in two *rPath*s would definitely lead a flow to two distinct physical paths. For flows between different host pairs, applying ports within a single *rPath* may route them to distinct physical paths, because their IP addresses differ. Inside a data center, one uplink leaf switch and one downlink spine switch determine a path, so Harp calculates the *rPath* set for the two switches. Besides, for the first hop in a path, the host selects the leaf switches in a round-robin fashion. With the port-based path control, Harp can map a flow to any path.

$$rPath := \{\, P_s \mid G_{k_1}^1 \cap \ldots G_{k_i}^i \cdots \cap G_{k_{M-1}}^{M-1} \,\}, 0 \le k_i \le N_i - 1 \tag{2}$$

Path pool implementation: According to the deterministic port-based path control, Harp identifies the physical paths between each pair of hosts and builds a path pool for them. Because a host can communicate with multiple hosts, the number of host pairs is enormous in large-scale data centers, which may incur a nontrivial storage overhead on hosts. Based on the deployment characteristics of Tencent Cloud, ECMP configurations for switches (including hashing algorithm and hashing seed) and the number of switches are generally the same on the same layer within a data center. Thus, the calculated UDP source port groups, i.e., G_k^i, for one switch can also apply to other switches on the same layer. In Fig. 5, the G_k^is for $Leaf_1$ to $Leaf_4$ are the same, while those for $Leaf_5$ to $Leaf_8$ are also considered the same. Therefore, for paths with the same leaf and spine switch configurations, Harp only calculates the set of *rPath*s once and uses it for all involved host pairs. For instance, in Fig. 5, there are eight paths between a host pair originating from one leaf switch. For each host pair that involves two leaf switches, we build eight *rPath*s and select one source port from each *rPath*. With a total of eight

source ports that Host1 sends over $eth0$ and $eth1$ iteratively, packets can span all 16 paths.

Empirically, cloud providers typically deploy VMs from the same tenant with locality, such as placing them in one data center. But communicating hosts can also reside in different data centers. The ECMP hash linearity holds only for intra-DC networks [31, 59], so the deterministic path control can't be applied to inter-DC traffic. We also observe that intersections of G_k^i may be empty on a small subset of switches. To compensate for the limitations, we enlarge the path pool with more ports. Specifically, the path pool is expanded by a multiple of 64 ports in our implementation. Besides, there are generally abundant source ports in the *rPath*, allowing us to select distinct ones. For instance, in Fig. 5, given the path pool size is 64, Harp will select eight ports from the eight *rPath*s, respectively. By increasing the variety of employed ports, the traffic can be distributed across more physical paths. Our production results demonstrate that Harp remains effective in this situation (e.g., DCI switch failure in §7).

3.3 Flow Mapping and Failure Handling

Flow Mapping: The Flow Mapping Module evenly distributes the VMs' traffic across the physical paths in the path pool to balance the paths' loads. In Harp, we build a Path Mapping Table that maps flows into paths between each host pair, which is represented by a NIC interface and UDP source port. During initialization, Harp identifies all available physical paths between a host pair and stores their NIC interface and UDP source port in a path table (PT). Fig. 6 shows a path table with 64 paths, and two adjacent paths have distinct uplink leaf switches, since $eth0$ and $eth1$ are organized in a round-robin fashion. For path remapping after failures, Harp builds an offset table (FT) that indexes the path table, which is changed dynamically. FT builds the mapping of flows to PT. The offset is used to bypass faulty paths in the event of network failures, and it stays 0 when the path is healthy. In the initialization, all paths are healthy, so all offsets in FT are initialized to 0, and FT and PT are directly mapped. If a flow is hashed to the ith item in FT, its first option is the ith path in PT. When a packet enters Harp, the Flow Mapping Module takes its inner 5-tuple as input and calculates the hash value to locate an item in FT. When a flow hashes to the ith slot in FT, it will look up its path in PT$[i+FT[i]]$.

Failure Handling: When a path suffers an outage, the Failure Handling Module reroutes the flows on the path to a healthy path. The module alters the offset by $(offset = offset+1)$ to switch to the following available path, i.e., the next slot whose *offset* is zero. For instance, in Fig. 6, when the path defined by *(eth0,srcport2)* fails, *offset2* is increased by 1, and *flow1* is directed to the following available path defined by *(eth1, srcport3)*. Upon detecting that the failed path has been restored, Harp reuses the repaired path by setting the corresponding $offset$ to 0. This design has two benefits. First, as the Failure Handling Module is aware of all paths' health states, one path-switch operation can reroute flows to a healthy path. Second, only the flows on the faulty path are rerouted, minimizing out-of-order issues in the flows during failure recovery. Note that multiple affected flows may be mapped to the same slot in PT after re-path, but flows between different host pairs are likely to be routed across distinct physical paths because their IP addresses differ. To improve load balancing for flows between a host pair, a random offset can be introduced during path selection in future work, i.e., $(offset = offset + rand)$.

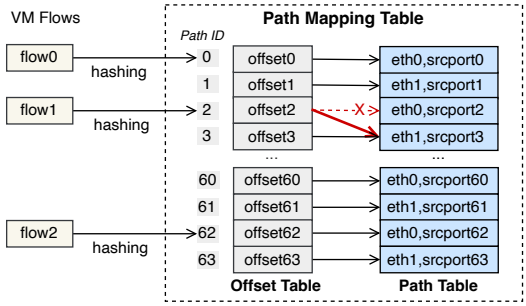

Figure 6: Mapping from flows to paths

4 Failure Detection

4.1 Basic Failure Detection Mechanism

The basic failure detection mechanism in Harp works as follows: the Failure Detection Module in the source host periodically sends inspection messages through the paths selected by the Flow Mapping Module, and the Failure Detection Module in the destination host records all the paths of inspection messages received. To minimize the network's bandwidth usage, the Failure Detection Module in the destination host periodically sends information about the recorded paths to the source host. After receiving and analyzing the replied path summary, the used paths whose inspection messages were not received by the destination host are considered failures.

We define *cycle* as a predetermined time interval, which is the granularity at which Harp monitors path states. From the start of a *cycle*, the Failure Detection Module in the source host sends the inspection message for the used path, where the inspection message contains two key fields, which are denoted as *path_id* and *inspection_period*. *path_id* is the index of the path through which the inspection message is routed in the path table (§3.3). Harp defines the time interval from receiving the last reply message to sending the inspection message as *inspection_period*, during which the used paths are pending confirmation of health states. The values of *inspection_period* in the inspection messages are accumulated as long as the reply message is not received. As shown in Fig. 7, at $t2$, the Failure Detection Module in the source host sends an inspection message through Path $p1$ and uses $t2-t1$

as the *inspection_period* because it receives the last reply message at $t1$. At $t3$, it sends another inspection message for Path $p2$, in which it uses $t3-t1$ as the *inspection_period*.

The Failure Detection Module in the destination host records the path ID when it receives the inspection message. When reaching the *cycle*, it sends a path summary for all active paths within a past period. We use *reply_processing_delay* to denote the time from receiving the last inspection message to sending the reply message in the destination host. In Fig. 7, the destination host receives the inspection message of Path $p2$ at $t4$, and it waits and sends a reply for Path $p2$ at the end of the *cycle* $t6$. In this case, $reply\_processing\_delay = t6-t4$. In addition to the path summary, the reply message includes a time that indicates the period within which the paths are active, and the time is the sum of *inspection_period* and *reply_processing_delay*. In Fig. 7, the reply is sent at $t6$ with $p2, t6-t4+t3-t1$, which means $p2$ is active within the past period $(t6-t4+t3-t1)$.

Harp defines *examination_period*, which is the time interval between two consecutive reply messages. Because Harp records path information in the period and uses the next reply to confirm path states, *examination_period* is the granularity of checking path states. The source host uses the recorded time in the reply message to analyze the path health states. In this case, $t6-t4+t3-t1$ approximates the time interval between two reply messages. On receiving the reply message, the source host checks backward the active paths within *examination_period*. If the replied path summary doesn't contain the used path, the path is treated as failed. In Fig. 7, after receiving the reply message at $t7$, the source host identifies $p1$ and $p3$ as failures since they are used within the range of *examination_period*, but the destination host does not receive their inspection messages. Instead, $p2$ is marked as healthy because its information is contained in the reply message.

This failure detection mechanism is robust and functions correctly even when reply messages are dropped. Reply messages may be lost due to network events such as congestion. Since the path used by the inspection message is generally different from the path that carries the reply message, the source host cannot know the real states of its active paths if the reply is lost. Each inspection message contains the time information of the last received reply. Therefore, whether the reply message is received or not, the destination host checks backward the active paths during the past period $inspection\_period + reply\_processing\_delay$. When the reply message at $t6$ is lost, the following inspection message that is sent at tx contains the inspection period of $tx-t1$. With the received inspection period, the destination host will keep recording the state of $p2$ and include its information in the following reply message. In this way, when the source host receives a reply at ty, it can know $p2$ is active during the examination period $[t1, ty]$.

To detect the restoration of faulty paths, the Failure Detection Module periodically inserts out-of-band inspection packets into faulty paths on a minute timescale. After the network operational team repairs the offending component, the replied path summary will include the repaired paths, then Harp can treat them as healthy and migrate traffic back to them. Since probing is infrequent and the number of failed paths is small, the overhead is negligible.

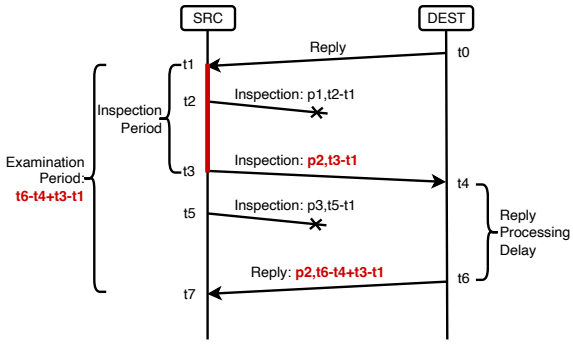

Figure 7: The basic failure detection mechanism

4.2 Failure Detection with Fault Tolerance

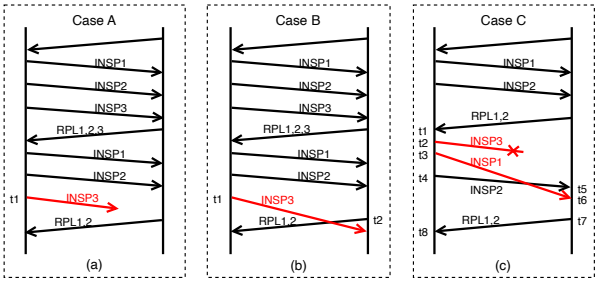

Figure 8: Deficiencies in the basic failure detection mechanism

When experiencing network congestion, packets may be lost, transmission delayed, or out of order. Although the basic mechanism can handle reply message loss, it may generate false alarms when facing other network issues. Fig. 8 depicts the behaviors of the basic mechanism in three typical cases, where *INSPi* denotes the inspection message for Path *i* and *RPLi* denotes the reply message with active Path *i*. 1) *Case A*: in Fig. 8(a), Path 3 is healthy, but its inspection message sent at $t1$ is lost. 2) *Case B*: Path 3 is healthy, but its inspection message arrives at the destination host after sending the reply message at $t2$ (Fig. 8(b)). This may happen when the transmission of inspection messages is delayed due to network congestion. When the subsequent inspection messages for Path 3 in *Case A* and *Case B* reach the destination at the following *cycle*s, it would be a **false positive** if Harp judges Path 3 as a failure. 3) *Case C*: *INSP*1 reaches the destination later than *INSP*2, and Path 3 suffers a failure between $t1$ and $t8$. This case is illustrated in Fig. 8(c), where the *examination_period* is only $t7-t6+t3-t1$, but Path 3 is not used by the source host within the time. Differently, Harp should be able to report

Path 3 as a faulty path in *Case C*; Otherwise, it would be a **false negative**.

To increase the robustness to deal with the common network issues, Harp requires the time to monitor path health states to span multiple *cycle*s to minimize the occurrence of false detection. Specifically, Harp extends *examination_period* to at least M successive *cycle*s (a *cycle* is set as C seconds), i.e., at least $M \times C$ seconds. When a reply message is sent, the destination host returns the path summary for the active paths within the time $max(M \times C, inspection\_period + reply\_processing\_delay)$. With the reply message being received at time t_k, the source host inspects the active paths during $[t_k - examination\_period, t_k]$. Algorithm 1 depicts the approach of preparing the received path IDs in the destination host. In line 2, we calculate the value of *examination_period*, which is the larger one between $M \times C$ and $inspection\_period + reply\_processing\_delay$. In lines 3-4, we calculate the range of the current *examination_period* as [*start*, *end*]. In lines 5-9, we obtain the active paths in the time range of [*start*, *end*]. At last, the algorithm returns the active path IDs and the *examination_period* to send to the host.

Algorithm 1 Prepare the received path IDs with fault tolerance

Input: *inspection_period*, *reply_processing_delay*
Output: *active_paths*, *examination_period*

```
1:  active_paths = ∅
2:  examination_period = max(M × C, inspection_period +
    reply_processing_delay)
3:  start = time() − examination_period
4:  end = time()
5:  for recorded paths do
6:      if start ≤ path.rx_time ≤ end then
7:          active_paths = active_path ⋃ path.id
8:      end if
9:  end for
10: return active_paths and examination_period
```

Fig. 9 illustrates an example where M equals three, Path 1's inspection message sent at $t2$ is lost, and Path 3's inspection message sent at $t5$ arrives at the destination after the reply message is sent. Before sending the reply message, the Failure Detection Module in the destination host obtains the recorded path IDs for time $M \times C$ ($3C$), which is greater than the sum of *inspection_period* ($t3 - t1$) carried in *INSP*4 and *reply_processing_delay* ($t6 - t4$). In this example, the reply message received at $t7$ contains active Path 1, 3, and 4, and *examination_period* is $3C$. As a result, the source host will not take Path 1 and Path 3 as faulty paths.

As we can see, the improved mechanism will not consider an active path as a failure unless the network issues occur continuously for M successive *cycle*s. Although it effectively reduces the possibility of false failure detection, it leads to a higher latency for detecting path failures. In the worst case, when a path suffers a failure immediately after *cycle* during which it is active, the failure can only be detected after M *cycle*s because the replies in the following $M-1$ *cycle*s will contain the path information. By controlling *examination_period*, Harp can adjust the trade-off between false failure detection and detection latency according to application scenarios.

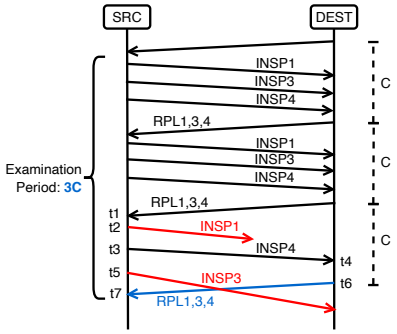

Figure 9: An example of failure detection with fault tolerance

4.3 Lightweight Implementation of Failure Detection

Embedding and Compression To minimize the impact on VM traffic, Harp proposes a lightweight implementation approach to perform failure detection. Instead of sending standalone packets for inspection and reply messages, Harp embeds probe information (28 bytes) into VM packets, thereby avoiding the overhead of encapsulating separate packets and enabling Harp to achieve high path coverage. In addition, considering most cloud traffic is bidirectional, the packet carrying the inspection message also involves the reply message in the reversed direction. Moreover, to reduce the number of bytes needed to store the recorded path IDs in the reply message, Harp compresses them into a bitmap, where each path ID consumes only one bit. Embedding 28-byte probe information into VM packets reduces the theoretical maximum network throughput by 1.87% at a 1500-byte MTU and 0.31% at a 9000-byte MTU for jumbo frames, whose overhead is trivial.

Sampling Sending inspection information in every packet is not only costly but also unnecessary. The information takes a non-trivial amount of extra network bandwidth when it is recorded in each sent packet. Besides, both the source and destination hosts need extra CPU cycles to process the information, which may impact the performance of the hosts. Therefore, Harp proposes to sample the sent packets and insert the inspection information in them. The inspection message for a path is only sent in one packet within a *cycle*.

When the reply message indicates a path failure, the path is switched for the next egress packet. However, when there is a huge delay between two subsequent packets in low-volume traffic, the reply message cannot be received in time to detect the failure. Moreover, low traffic volume also results in a low physical path utilization ratio between two hosts. Since the

health states of the unused paths are unknown, the switched new path may still be faulty. Consequently, multiple path switches may be required to recover the network for affected flows. These two main factors may prolong the recovery process for cases with low traffic volumes.

5 Deployment Experiences

5.1 End Host Problems

We find that the deployment of Harp in Tencent Cloud causes false detection in Cases 1 and 2, and that Harp can help localize the misconfigured NIC on the end host in Case 3.

Case 1: Dropped probe packets. Some end hosts occasionally drop received packets in production. Specifically, when experiencing a bursty increase in ingress traffic, the NIC's receive buffers can rapidly run out, and subsequent incoming packets will be discarded. If packets with inspection messages are dropped on destination hosts, Harp may consider outages to have occurred, since it can't distinguish between in-network and end-host packet loss. In this situation, re-path is unnecessary, and inspection messages on the same path are unlikely to be dropped successively. Thus, Harp embeds the number of NIC packet losses in the probe information. When NIC packet losses are observed, Harp only reroutes flows on the path after it has been judged as failed by consecutive dropped inspection messages.

Case 2: Probe packets queuing delay. We observe false alarms when the CPU usage rate on the end-host spikes to a high level. CPU overload can increase the waiting time of packets in NIC ring buffers, and delaying the processing of Harp's probe information will introduce an offset to the examination period. For instance, in Fig. 7, if the reply message arriving at the source host at $t7$ is processed at $(t7+4ms)$ after experiencing a delay, the time window of the examination period will be forward shifted 4 ms $([t7-(t6-t4+t3-t1)+4ms, t7+4ms])$. But the replied path summary doesn't contain the information within the 4 ms offset, thus causing a false detection. Commercial NICs don't provide hardware timestamps, so the precise queuing delay can't be identified. To mitigate the problem, when processing a reply message, Harp will not examine health states for paths active within the trailing milliseconds of the examination period. In the above example, the source host will not check active paths during $[t7, t7+4ms]$.

Case 3: Localizing misconfigured NIC. Early this year, many hosts simultaneously detected failed paths, and the monitoring logs showed that they communicated with the same destination host. With the information from Harp, the operational team promptly identified that the NIC on the destination host was misconfigured, causing one of the two interfaces to drop all received packets. Moreover, by analyzing patterns of path health bitmaps in the monitoring logs, Harp can assist the operational team in locating faulty devices. For example, patterns of 0x0101...01 and 0x1010...10 indicate that one of the two uplink leaf switches or NIC interfaces has failed, based on the arrangement of paths in the path table (§3.3).

5.2 Challenges in Deploying Harp at Scale

Dynamic routing in DCN: Maintenance operations are performed routinely in data centers, such as adding or decommissioning PoDs. Border Gateway Protocol (BGP) [1, 40] reacts to changes in network topologies and updates ECMP groups on switches. In the event of a link or network device failure, BGP route convergence can also result in routing changes. The dynamic routing characteristic of DCN requires Harp to maintain validity for employed UDP ports. First, we fix the ECMP configurations to prevent ECMP results from changing after switch reboots. Second, we deploy an offline tool to periodically obtain ECMP results; upon detecting changes, Harp recalculates *rPath*s and updates used UDP ports in the path pool. Maintenance updates are infrequent and planned, but network failures occur unpredictably. We observe that the daily failed link ratio across all data centers is extremely low (<0.0002%), and the operational team can quickly repair offending components, allowing outdated ports to resume validity soon. Thus, the offline tool updates UDP ports on a monthly schedule rather than reactively after failures.

Probe cycle setting: The primary consideration in setting the *cycle* is failure recovery efficiency. Although workloads in cloud environments are diverse, TCP-based workloads generally have a higher latency requirement than UDP-based workloads. Based on this empirical observation, Harp aims to restore network connectivity before the first TCP RTO (typically 200 ms). The detection latency in Harp is controlled by *examination_period*, which is at least $M \times C$ (§4.2). Thus, the *cycle* is selected to keep $M \times C$ below one TCP RTO. The *cycle* should be greater than the Round-Trip Time, and a smaller *cycle* can incur higher overhead. Considering the three factors, the *cycle* is on the order of multiple milliseconds in production systems.

Path probe coverage: When the path utilization ratio is low, Harp may redirect flows to undetected faulty paths due to a lack of traffic to probe (§4.3). To increase the path probe coverage, Harp temporarily redirects a VM packet with probe information to an unused path if the path utilization ratio is below a threshold. We also limit the count of redirected packets to a low value to mitigate out-of-order issues caused by redirection.

5.3 Planned Optimizations

Queuing latency measurement with SmartNIC: SmartNICs that support Harp are undergoing deployment in Tencent Cloud. Hardware timestamps are readily supported on SmartNICs, enabling the measurement of precise queuing delay. By removing the exact delay within the examination period, Harp

can address false detections due to packet queuing (§5.1). It also enables accurate measurement of the transmission latency for each path. Thus, Harp can estimate the traffic load from the latency and select paths based on load, thereby optimizing load balancing.

Path control in switches with different ECMP: Harp currently doesn't support cloud gateway traffic, and switches in our gateway clusters don't support uniform ECMP configurations. We plan to employ multiple ECMP groups on switches to control paths [31], i.e., $<DSCP, hash(outer\_srcip, outer\_dstip)>$. First, we configure each switch to ensure distinct output ports at corresponding offsets across ECMP groups. For example, the port lists of two ECMP groups are $[P0, P1, ..., PN-1]$ and $[P1, ..., PN-1, P0]$. Second, the *DSCP* in each packet's IP header selects one ECMP group, and $hash(outer\_srcip, outer\_dstip)$ maps the packet to an offset within the ECMP group's port list. Thus, for packets between a host pair, because their offsets are the same, toggling *DSCP*s can direct them to distinct output ports.

6 Analysis of Failure Handling with Harp in Production Systems

6.1 Failure of Core Switches

Early in 2024, multiple interfaces of a core switch were down due to hardware errors. The network didn't recover until the core switch rebooted after tens of seconds. According to logs of our network operational system, the traffic carried by the faulty core switch accounted for only 1% of all traffic, while 0.7% of packets were lost during the failure. The failure caused timeouts in many Redis and COS instances deployed on servers without Harp. For servers deployed with Harp, no user-visible failures were reported. Around 55% of flows were restored from the failure within 0.4 s, and 33% of flows took 0.5 s to 1 s to recover. As we can see, Harp can significantly shorten the duration of failure. The remaining 12% flows require seconds of failure recovery time. According to the monitoring logs, there was no network traffic on the corresponding servers during the few seconds when the failure occurred. Harp started detecting failures only after the VMs resumed their traffic (§4.3). This is why it took Harp much longer to recover the flows.

6.2 Failure of DCI Switches

Case 1: A DCI switch upgrade failed. In January 2025, a DCI switch failed during an upgrade. A small number of paths failed for tens of seconds, and the problem was fixed after rebooting the switch. The failure recovery time with Harp is shown in Fig. 10(a) (the blue line). 90% of the influenced flows were recovered within 400 ms, and 99% of flows were recovered within 1 s. This demonstrates the effectiveness of Harp's fast failure recovery in production systems.

Case 2: A DCI switch stopped forwarding. In 2024, a switch in the DCI network stopped forwarding, and traffic routed through it was silently dropped, causing the majority of paths connecting the two data centers to be unavailable. The failure lasted a few minutes, while Harp helped to recover the traffic in a shorter time. The CDF of failure recovery time with Harp (the orange line) is depicted in Fig. 10(a). The failure recovery time ranged from 16 ms to 4.6 s. About 43% of influenced flows were restored within 1 s, and 99% of influenced flows were restored within 4.5 s.

In the two cases above, Case 1 has a much lower recovery time. The reason is as follows. Fig. 10(b) compares the path utilization ratios for the two cases, where the path utilization in Case 1 was much higher than in Case 2 because of the corresponding traffic pattern. Specifically, for Case 1, 90% of host pairs used 90% of paths, while only 30% of paths are used for 90% of host pairs in Case 2. Consequently, Harp, in Case 1, knows the health states of more physical paths than in Case 2. Fig. 10(c) depicts the distributions of path-switching numbers for the two cases. For Case 1, 85% of flows successfully switched to healthy paths with only one switching operation, while at most two switching operations were demanded for the remaining 15% host pairs. Instead, in Case 2, 57% of the influenced flows performed one path switch, but 43% succeeded in finding healthy paths with 2 to 4 switches. Furthermore, the failure in Case 2 affected most paths, increasing the likelihood that Harp rerouted the affected flows to faulty paths.

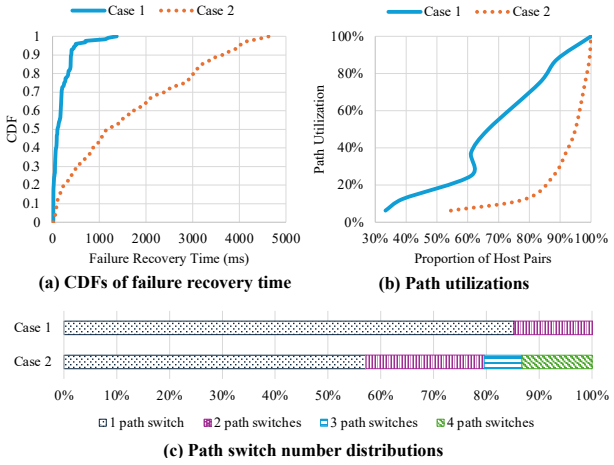

Figure 10: Recovery on DCI Switch outages with Harp

7 Evaluation

We evaluate Harp in production systems, where hundreds of thousands of servers are deployed with Harp, involving a variety of cloud workloads running in VMs, such as storage and security. The evaluation is performed from four aspects: (1) effectiveness of failure detection and recovery, (2) failure recovery efficiency, (3) cloud application recovery, and (4) overhead. We use the physical network diagnosis approach

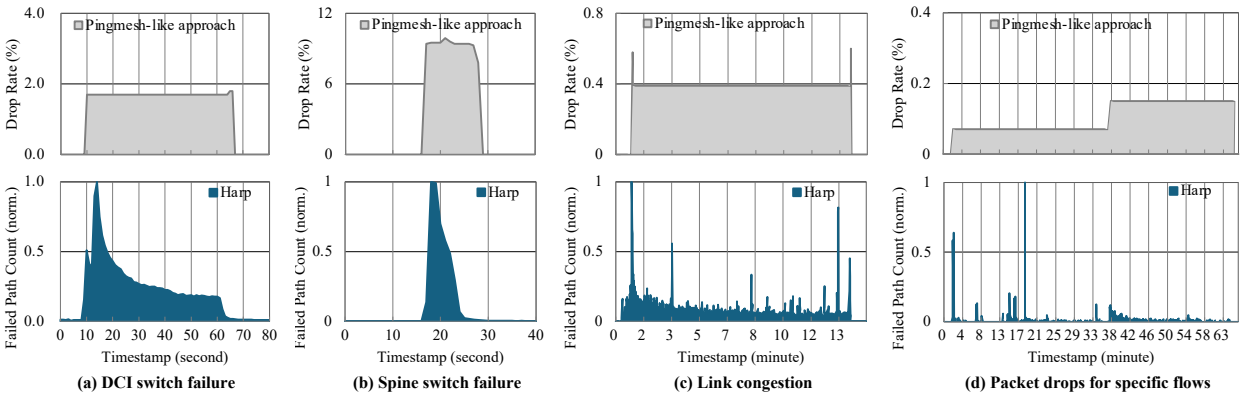

Figure 11: The monitoring of network availability in production

in Tencent Cloud (§2.2) as the baseline, which monitors the drop rate of probe packets using a Pingmesh-like approach.

Effectiveness: Each server is equipped with a tool that monitors the count of failed paths identified by Harp every second. An increase in the count indicates the occurrence of a network failure, while a decrease indicates recovery of the VPC network's connectivity. Fig. 11 depicts the monitoring results over a period during which four real-world network failures occur. The failed path counts are normalized to the maximum value during the time. In Fig. 11(a), a DCI switch stops forwarding at 10 s, and the Pingmesh-like approach has a packet drop rate of approximately 1.9% for about one minute. In Fig. 11(b), a spine switch fails at 16 s, and the routing is automatically converged after 10 s. The Pingmesh-like approach observes a 9% packet drop rate during the failure. For Harp, the counts of identified failed paths increase as soon as the failures occur, and then gradually decrease in both cases. The results demonstrate that Harp detects the outages rapidly, and the reduction in failed path counts indicates that Harp successfully reroutes affected flows to healthy paths.

Fig. 11(c) and (d) illustrate the results of two gray failures. A small subset of links is congested due to an error on the wavelength division board of a DCI switch in Fig. 11(c). In Fig. 11(d), the system software of a leaf switch encounters an error, causing packet drops for specific flows. In both cases, the packet drop rates of the Pingmesh-like approach are too low to alert. In contrast, because Harp has higher coverage, it promptly identifies failed paths for affected flows; the failed path count is swiftly reduced to a low level, indicating that Harp effectively helps flows to bypass faulty paths. In Fig. 11(c), although Harp redirects VM traffic to non-congested paths, a high volume of non-VM traffic still exists on the congested links. Thus, the Pingmesh-like approach continues to experience packet drops. Moreover, in all four cases, Harp continuously identifies failed paths before the network operational system repairs the failures. This is because Harp starts detecting failures as long as there is VM traffic.

Failure recovery efficiency: Switch failures account for most of the outages in our data centers (§2.2). Over a six-month monitoring period, eight switch failures with broad impacts have occurred in production systems, including spine, core, and DCI switch failures. For these failures, we compare the recovery latency of Harp with that of the network operational system's failure handling approach. The results are shown in Table 1. Without Harp, the network operational system takes 10 s to 180 s to handle the network failures. In contrast, the P25, P50, and P75 latencies that Harp takes to detect failed paths and reroute affected flows to healthy ones are at most 65.5 ms, 97.2 ms, and 206.5 ms, respectively. The P95 latencies of Harp don't exceed 1.28 s. As we can see, 50% of traffic recovers from failures within half of the 200ms-RTO, while 75% recovers in approximately one 200ms-RTO. Most of the traffic can bypass failed paths on a sub-second timescale. Compared with the network operational system's approach, Harp significantly reduces the VPC network's outage time by 78.71%-99.97%. The tail latency (P99) ranges from 0.78 s to 3.54 s, and low traffic volume is the main cause of a long tail latency. In addition to failures with broad impacts, we also look at the overall behavior of Harp over the monitoring period. Specifically, the average failure recovery time with Harp is 295 ms, and Harp reduces the VPC network's downtime by approximately 99.9%.

Cloud application recovery: RTO and failed TCP connection attempts are two critical TCP metrics for monitoring network availability; increments in their counts suggest high network latency or packet loss. In our data centers, some servers do not enable Harp because they are undergoing a temporary deployment phase. We collect these two TCP metrics in VMs running on the servers with and without Harp. The comparisons of their increments in the event of real-world network failures are depicted in Fig. 12. As we can see, the increments of both metrics in VMs are significantly reduced with Harp. The results indicate that Harp effectively improves VPC network availability, enabling cloud applications to recover from network failures more quickly.

Overhead: In cloud environments, resources allocated to the infrastructure layer on each server are limited. Low overhead is a prerequisite for deploying Harp at scale. Harp performs in-band detection, thus eliminating additional network

Table 1: Comparisons of failure recovery time in real-world network failures

No	Network Failure Cases	W/O Harp	Harp P25	Harp P50	Harp P75	Harp P95	Harp P99
1	Spine Switch Hardware Error	10 s	31.0 ms	71.4 ms	206.3 ms	0.56 s	2.05 s
2	Spine Switch Hardware Error	12 s	35.3 ms	48.0 ms	56.4 ms	0.14 s	0.78 s
3	Spine Switch Hardware Error	15 s	30.2 ms	74.3 ms	205.5 ms	0.42 s	3.19 s
4	Core Switch Hardware Error	25 s	43.7 ms	97.2 ms	206.5 ms	0.64 s	3.03 s
5	Core Switch Abnormal Shutdown	51 s	45.0 ms	65.9 ms	206.1 ms	1.28 s	3.0 s
6	Core Switch Hardware Error	103 s	34.6 ms	49.5 ms	57.2 ms	0.36 s	1.25 s
7	Core Switch Hardware Error	180 s	65.5 ms	65.5 ms	65.5 ms	0.65 s	3.54 s
8	DCI Switch Hardware Error	20 s	33.6 ms	80.7 ms	205.5 ms	0.44 s	1.48 s

bandwidth overhead. Harp is implemented as a dynamic library and works alongside the virtual switch. The CPU is primarily consumed by periodically embedding probe information into VM packets; the memory overhead comes from saving path health states. Fig. 13 depicts the average resource usage on 80K+ servers over 15 days. The CPU usage rate and memory consumption are below 1.2% and 13.5 MB, respectively. The results reveal that Harp has negligible overhead, and its resource consumption is acceptable for the vast majority of servers in production data centers.

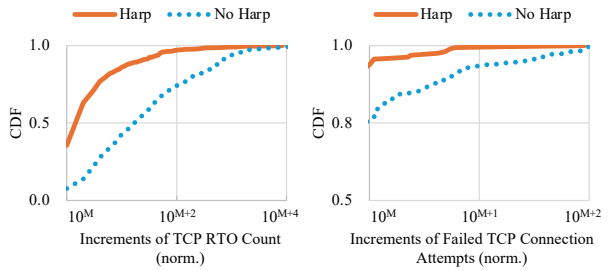

Figure 12: Failure recovery in the application level

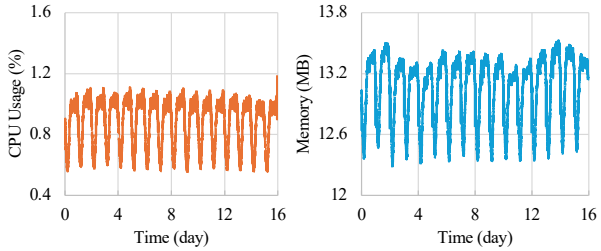

Figure 13: CPU and memory usages by Harp

8 Related Work

Localization-then-mitigation approaches: There is a spectrum of work [4, 11, 17, 18, 28, 34, 42, 47, 50, 56, 58, 62, 63] that collect system metrics to localize faulty components, and then take actions to alleviate failures. Since localizing failures is time-consuming, those approaches generally take tens of seconds or even minutes to mitigate failures. For example, BIAN [50] uses LLM to localize failures and analyze root causes. ByteTracker [18] reduces the latency to several seconds by utilizing the ERSPAN feature on switches. To further reduce the latency, R-Pingmesh [28] sends out-of-band probe packets at a millisecond-level interval in RoCE networks, but this probe frequency can incur considerable overhead in ultra-large data centers.

Multipath approaches: [10, 22, 23, 32, 43, 45, 53] leverage multipath in data centers and achieve fast failover via randomly rerouting. Nevertheless, because rerouting is random, the recovery latencies of those approaches are unpredictable. Harp differs from proactively monitoring path states, thereby guaranteeing efficient recovery. ZooRoute [46] actively sends probe packets for host pairs and bypasses failures within seconds. Harp employs an efficient in-band failure detection mechanism, enabling sub-second recovery.

BGP converges after switch/link failures. But empirically, it can take seconds when the switch CPU has limited processing capacity and the routing table is large. Harp complements BGP in scenarios where BGP is less efficient and in handling failure types beyond BGP's capabilities, e.g., gray failures. Some hardware-based approaches [5, 13, 21, 26, 51] require support from special hardware, and SDN-based approaches [14, 29, 44] require changes to network routing. In contrast, Harp can be deployed directly on existing infrastructures. Virtual network diagnosis approaches [12, 30, 41, 52, 61] probe virtual paths between VM/container pairs. Harp proposes monitoring physical paths between host pairs, thereby lowering computational and state storage overhead by significantly reducing the number of probing paths. ECMP-based approaches [31, 59] explore switch properties to precisely control traffic paths that are complementary to Harp. Harp's failure detection mechanism is also applicable to IPv6 networks by employing IPv6's path control approaches, such as Segment Routing [49] and Flow Labels.

9 Conclusion

Harp has proven to be an effective and lightweight network failure detection and recovery mechanism for VPC networks in Tencent Cloud. By monitoring the health states of physical paths, Harp can essentially reroute flows to healthy paths on a sub-second timescale. Harp does not rely on specific hardware support and is transparent to cloud applications. These features are critical for meeting the stringent, specific management and performance requirements of modern cloud data centers.

References

[1] Anubhavnidhi Abhashkumar, Kausik Subramanian, Alexey Andreyev, Hyojeong Kim, Nanda Kishore Salem, Jingyi Yang, Petr Lapukhov, Aditya Akella, and Hongyi Zeng. Running {BGP} in data centers at scale. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 65–81, 2021.

[2] 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.

[3] Hamada Alshaer. An overview of network virtualization and cloud network as a service. *International Journal of Network Management*, 25(1):1–30, 2015.

[4] Behnaz Arzani, Selim Ciraci, Luiz Chamon, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Boon Thau Loo, and Geoff Outhred. 007: Democratically finding the cause of packet drops. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 419–435, 2018.

[5] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. Pint: Probabilistic in-band network telemetry. 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 662–680, 2020.

[6] Josiah Carlson. *Redis in action*. Simon and Schuster, 2013.

[7] Tencent Cloud. Tencent cloud file storage. https://www.tencentcloud.com/products/cfs?from_qcintl=122040102, 2025.

[8] Tencent Cloud. Tencent cloud object storage. https://www.tencentcloud.com/products/cos?from_qcintl=122040101, 2025.

[9] Tencent Cloud. Tencentdb for redis. https://www.tencentcloud.com/products/crs?from_qcintl=122050301, 2025.

[10] 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.

[11] Facebook Engineering. Netnorad: Troubleshooting networks via end-to-end probing, February 2016. Accessed: 2025-09-02.

[12] 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.

[13] Chen Feng, Yibo Zhang, Chen Zhang, and Beichuan Zhang. Flow event telemetry on programmable data plane. In *Stanford CS244: Foundations of Networking*, 2022.

[14] Andrew D Ferguson, Steve Gribble, Chi-Yao Hong, Charles Killian, Waqar Mohsin, Henrik Muehe, Joon Ong, Leon Poutievski, Arjun Singh, Lorenzo Vicisano, et al. Orion: Google's {Software-Defined} networking control plane. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 83–98, 2021.

[15] 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.

[16] Haryadi S Gunawi, Mingzhe Hao, Riza O Suminto, Agung Laksono, Anang D Satria, Jeffry Adityatama, and Kurnia J Eliazar. Why does the cloud stop computing? lessons from hundreds of service outages. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 1–16, 2016.

[17] 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.

[18] 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.

[19] Peng Huang, Chuanxiong Guo, Lidong Zhou, Jacob R Lorch, Yingnong Dang, Murali Chintalapati, and Randolph Yao. Gray failure: The achilles' heel of cloud-scale systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 150–155, 2017.

[20] Internet Engineering Task Force. VXLAN GPE: Generic Protocol Extension for VXLAN. https://www.ietf.org/archive/id/draft-ietf-nvo3-vxlan-gpe-12.html, Accessed on 2021.

[21] Chenhao Jia, Tian Pan, Zizheng Bian, Xingchen Lin, Enge Song, Cheng Xu, Tao Huang, and Yunjie Liu. Rapid detection and localization of gray failures in data centers via in-band network telemetry. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.

[22] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 149–160, 2014.

[23] J. Kim, S. Kohler, M. Handley, and D. Thaler. Tcp extensions for multipath operation with multiple addresses. https://www.rfc-editor.org/rfc/rfc6824.html#page-5, January 2013.

[24] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[25] Charles E Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

[26] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. Lossradar: Fast detection of lost packets in data center networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 481–495, 2016.

[27] Zhenhua Li, Cheng Jin, Tianyin Xu, Christo Wilson, Yao Liu, Linsong Cheng, Yunhao Liu, Yafei Dai, and Zhi-Li Zhang. Towards network-level efficiency for cloud storage services. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 115–128, 2014.

[28] Kefei Liu, Zhuo Jiang, Jiao Zhang, Shixian Guo, Xuan Zhang, Yangyang Bai, Yongbin Dong, Feng Luo, Zhang Zhang, Lei Wang, et al. R-pingmesh: A service-aware roce network monitoring and diagnostic system. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 554–567, 2024.

[29] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas Anderson. F10: A {Fault-Tolerant} engineered network. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 399–412, 2013.

[30] Wei Liu, Kun Qian, Zhenhua Li, Tianyin Xu, Yunhao Liu, Weicheng Wang, Yun Zhang, Jiakang Li, Shuhong Zhu, Xue Li, et al. Skeletonhunter: Diagnosing and localizing network failures in containerized large model training. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 527–540, 2025.

[31] 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.

[32] 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.

[33] 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.

[34] Pooria Namyar, Arvin Ghavidel, Daniel Crankshaw, Daniel S Berger, Kevin Hsieh, Srikanth Kandula, Ramesh Govindan, and Behnaz Arzani. Enhancing network failure mitigation with {Performance-Aware} ranking. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 335–357, 2025.

[35] 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.

[36] Rahul Potharaju and Navendu Jain. When the network crumbles: An empirical study of cloud network failures and their impact on services. In *Proceedings of the 4th annual Symposium on Cloud Computing*, pages 1–17, 2013.

[37] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu,

Rui Miao, et al. Alibaba hpn: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 691–706, 2024.

[38] Paul Quinn, Uri Elzur, and Carlos Pignataro. Network service header (nsh). Technical report, 2018.

[39] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath tcp. *ACM SIGCOMM Computer Communication Review*, 41(4):266–277, 2011.

[40] Yakov Rekhter, Tony Li, and Susan Hares. Rfc 4271: A border gateway protocol 4 (bgp-4), 2006.

[41] 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.

[42] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C Snoeren. Passive realtime datacenter fault detection and localization. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 595–612, 2017.

[43] 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.

[44] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM computer communication review*, 45(4):183–197, 2015.

[45] Arjun Singhvi, Nandita Dukkipati, Prashant Chandra, Hassan MG Wassel, Naveen Kr Sharma, Anthony Rebello, Henry Schuh, Praveen Kumar, Behnam Montazeri, Neelesh Bansod, et al. Falcon: A reliable, low latency hardware transport. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 248–263, 2025.

[46] Xiaoqing Sun, Xionglie Wei, Xing Li, Ju Zhang, Bowen Yang, Yi Wang, Ye Yang, Yu Qi, Le Yu, Chenhao Jia, et al. Zooroute: Enhancing cloud-scale network reliability via overlay proactive rerouting. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 1251–1253, 2025.

[47] Cheng Tan, Ze Jin, Chuanxiong Guo, Tianrong Zhang, Haitao Wu, Karl Deng, Dongming Bi, and Dong Xiang. {NetBouncer}: Active device and link failure localization in data center networks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 599–614, 2019.

[48] Dave Thaler and C Hopps. Multipath issues in unicast and multicast next-hop selection. Technical report, 2000.

[49] Pier Luigi Ventre, Stefano Salsano, Marco Polverini, Antonio Cianfrani, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo, and Francois Clad. Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results. *IEEE Communications Surveys & Tutorials*, 23(1):182–221, 2020.

[50] Chenxu Wang, Xumiao Zhang, Runwei Lu, Xianshang Lin, Xuan Zeng, Xinlei Zhang, Zhe An, Gongwei Wu, Jiaqi Gao, Chen Tian, et al. Towards llm-based failure localization in production-scale networks. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 496–511, 2025.

[51] Weitao Wang, Xinyu Crystal Wu, Praveen Tammana, Ang Chen, and TS Eugene Ng. Closed-loop network performance monitoring and diagnosis with {SpiderMon}. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 267–285, 2022.

[52] Zhe Wang, Huanwu Hu, Linghe Kong, Xinlei Kang, Qiao Xiang, Jingxuan Li, Yang Lu, Zhuo Song, Peihao Yang, Jiejian Wu, et al. Diagnosing application-network anomalies for millions of {IPs} in production clouds. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 885–899, 2024.

[53] 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.

[54] Timothy Wood, Prashant J Shenoy, Alexandre Gerber, Jacobus E van der Merwe, and Kadangode K Ramakrishnan. The case for enterprise-ready virtual private clouds. In *HotCloud*, 2009.

[55] Jiyi Wu, Lingdi Ping, Xiaoping Ge, Ya Wang, and Jianqing Fu. Cloud storage as the infrastructure of cloud computing. In *2010 International conference on intelligent computing and cognitive informatics*, pages 380–383. IEEE, 2010.

[56] Xin Wu, Daniel Turner, Chao-Chih Chen, David A Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. Netpilot: Automating datacenter network failure mitigation. In *Proceedings of the ACM SIGCOMM 2012*

conference on Applications, technologies, architectures, and protocols for computer communication, pages 419–430, 2012.

[57] Zhang Xu, Haining Wang, and Zhenyu Wu. A measurement study on co-residence threat inside the cloud. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 929–944, 2015.

[58] Bo Yang, Huanwu Hu, Yifan Li, Yunguang Li, Xiangyu Tang, Bingchuan Tian, Gongwei Wu, Jianfeng Xu, Xumiao Zhang, Feng Chen, et al. Skynet: Analyzing alert flooding from severe network failures in large cloud infrastructures. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 512–526, 2025.

[59] 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.

[60] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. Wcmp: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*, pages 1–14, 2014.

[61] 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.

[62] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y Zhao, et al. Packet-level telemetry in large datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 479–491, 2015.

[63] Danyang Zhuo, Monia Ghobadi, Ratul Mahajan, Klaus-Tycho Förster, Arvind Krishnamurthy, and Thomas Anderson. Understanding and mitigating packet corruption in data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 362–375, 2017.