Diagnosing Application-network Anomalies for Millions of IPs in Production Clouds

Zhe Wang, *Shanghai Jiao Tong University;* Huanwu Hu, *Alibaba Cloud;* Linghe Kong, *Shanghai Jiao Tong University;* Xinlei Kang and Teng Ma, *Alibaba Cloud;* Qiao Xiang, *Xiamen University;* Jingxuan Li and Yang Lu, *Alibaba Cloud;* Zhuo Song, *Shanghai Jiao Tong University and Alibaba Cloud;* Peihao Yang, *Alibaba Cloud;* Jiejian Wu, *Shanghai Jiao Tong University;* Yong Yang and Tao Ma, *Alibaba Cloud;* Zheng Liu, *Alibaba Cloud and Zhejiang University;* Xianlong Zeng and Dennis Cai, *Alibaba Cloud;* Guihai Chen, *Shanghai Jiao Tong University*

https://www.usenix.org/conference/atc24/presentation/wang-zhe

This paper is included in the Proceedings of the 2024 USENIX Annual Technical Conference.

July 10–12, 2024 • Santa Clara, CA, USA

978-1-939133-41-0

Open access to the Proceedings of the 2024 USENIX Annual Technical Conference is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Diagnosing Application-network Anomalies for Millions of IPs in Production Clouds

Zhe Wang[1], Huanwu Hu[2], Linghe Kong[1], Xinlei Kang[2], Teng Ma[2], Qiao Xiang[3], Jingxuan Li[2], Yang Lu[2]

Zhuo Song[1,2], Peihao Yang[2], Jiejian Wu[1], Yong Yang[2], Tao Ma[2], Zheng Liu[2,4]

Xianlong Zeng[2], Dennis Cai[2], Guihai Chen[1]

[1]*Shanghai Jiao Tong University,* [2]*Alibaba Cloud,* [3]*Xiamen University,* [4]*Zhejiang University*

Abstract

Timely detection and diagnosis of application-network anomalies is a key challenge of operating large-scale production clouds. We reveal three practical issues in a cloud-native era. First, impact assessment of anomalies at a (micro)service level is absent in currently deployed monitoring systems. Ping systems are oblivious to the "actual weights" of application traffic, *e.g.*, traffic volume and the number of connections/instances. Failures of critical (micro)services with large weights can be easily overlooked by probing systems under prevalent network jitters. Second, the efficiency of anomaly routing (to a blamed application/network team) is still low with multiple attribution teams involved. Third, collecting fine-grained metrics at a (micro)service level incurs considerable computational/storage overheads, however, is indispensable for accurate impact assessment and anomaly routing.

We introduce the application-network diagnosing (AND) system in Alibaba cloud. AND exploits the single metric of TCP retransmission (*retx*) to capture anomalies at (micro)service levels and correlates applications with networks end-to-end. To resolve deployment challenges, AND further proposes three core designs: (1) a collecting tool to perform filtering/statistics on massive *retxs* at the (micro)service level, (2) a real-time detection procedure to extract anomalies from 'noisy' *retxs* with millions of time series, (3) an anomaly routing model to delimit anomalies among multiple target teams/scenarios. AND has been deployed in Alibaba cloud for over three years and enables minute-level anomaly detection/routing and fast failure recovery.

1 Introduction

Cloud regions host the core business systems of Alibaba and serve billions of customers worldwide [21, 32]. The key challenge of operating Alibaba cloud is timely detection and diagnosis of application-network anomalies, to ensure service level agreements (SLAs) and avoid customer, reputation, and revenue losses caused by SLA violations [56].

Generally, we define application-network anomalies at three levels according to their severity. Firstly, applications exhibit performance jitters while the expected SLAs of customers are satisfied. Secondly, applications experience severe performance declines in throughput, latency, job completion time, *etc.*, and the SLAs of customers are violated. Thirdly, services are unavailable and anomalies upgrade to failures.

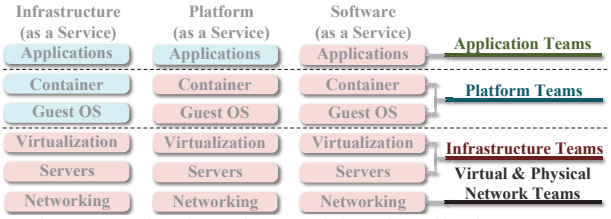


Figure 1: Cloud service model and attribution teams.

In the following discussion, we focus on severe application-network anomalies of levels 2 and 3. Our clouds provide a multi-layer service model including software, platform and infrastructure layers (Figure 1). Anomalies may happen at each layer in the end-to-end path (Figure 2). To ensure the stable operation of each layered component, each attribution team has built abundant monitoring systems at end hosts [10,23,49], virtual networks [15,38,57], physical networks [11,22,44,47,56] and correlate connections with physical links/paths [9,39,43].

We observe that currently deployed monitoring systems still face issues of impact assessment, anomaly routing and large operation overheads in a cloud-native era. Next, we elaborate on these practical issues in production.

Impact Assessment at (Micro)services Level. The clouds deploy (micro)services of various applications densely at shared hosts. We still lack a monitoring system for accurate impact assessment of anomalies at a (micro)service granularity. For example, currently deployed network probing systems may achieve a full-mesh probing at the host level [22, 57]. As shown in Figure 3, they probe each node/link equivalently and cannot capture actual weights for nodes/links of critical (micro)services, *e.g.*, in-memory database [8], proxy and load balancing [51]. As a result, probing results can be easily covered by noises of prevalent network jitters in production environments. We observe many cases where failures on nodes/links of critical services are overlooked by probings. Other monitors at the platform or infrastructure layer face a similar issue — they cannot tell whether or how many (micro)services are affected even though anomalies are detected.

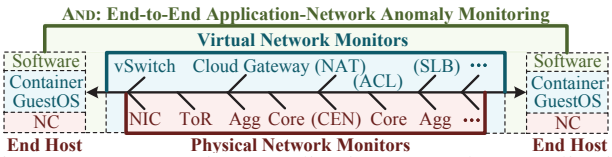


Figure 2: AND monitors application-network anomalies in the end-to-end path.

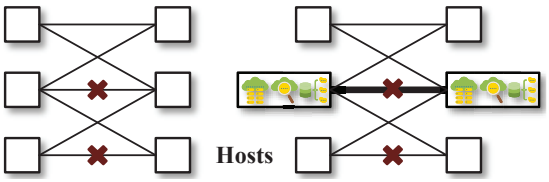


Figure 3: Probing systems (left) regard each node/link as equivalent, while partial nodes/links carrying traffic loads of critical (micro)services should have larger weight (right).

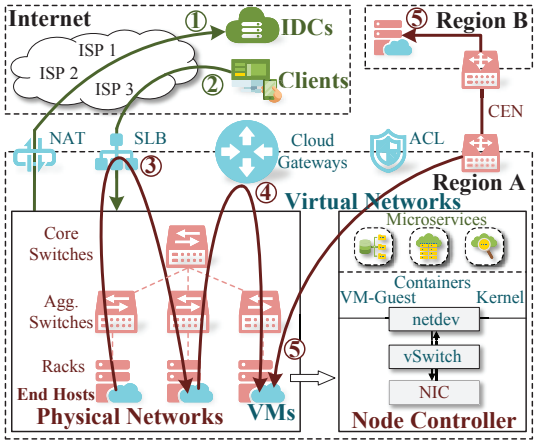


Figure 4: Overview of cloud traffic.

Anomaly Routing to Attribution Teams. The root cause of application anomalies is usually attributed to dependent (micro)services or shared infrastructure with a (micro)service-based architecture [17, 18]. Currently, the efficiency of anomaly routing to a blamed layer is still low, with multiple attribution teams and complex traffic scenarios involved (Figures 1 and 4). First, each application team reports its failure and inquires about other related teams respectively, leading to high communication costs. Second, correlating application metrics with problematic (micro)services or network events is non-trivial due to the inconsistent data format and semantics [20, 50]. The essential issue lies in that operation teams and specific monitoring tools focus on their problem domains (software, operating systems, networks). The clouds lack a unified diagnostic mechanism to correlate layered components and route anomalies to attribution teams.

Full Coverage with Low Overheads. Collecting connection-level metrics [39, 43] is indispensable for accurate impact assessment and anomaly routing, however, it incurs large capital/operating expenses (CapEx/OpEx) as an always-on service. Considering million-level (micro)service instances and billion-level connections, the monitoring system brings considerable computational/storage overheads (Figures 5 and 6).

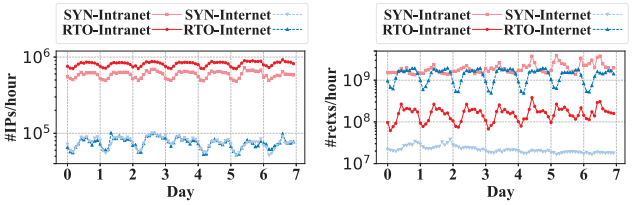
To tackle the above issues, we introduce the experience in designing and deploying the Application-Network Diagnosing (AND) system in Alibaba cloud. AND exploits TCP retransmissions (*retxs*) to extract anomalies with low overheads, capture problems at a (micro)service level, and correlate applications with platform/infrastructure layers end-to-end. Existing works [9, 10] also collect TCP *retxs* and statistics at end hosts for anomaly detection and diagnosis. We observe several deployment challenges that hinder the direct usage of these systems in a cloud-native environment.

Deployment Challenges. Firstly, applications exhibit massive connections and *retxs* at (micro)service/container level ($10 \sim 10^5$/min) in production clouds. As shown in Figure 6, collecting all connection-level metrics incurs unacceptable CPU/storage overheads. Collecting and analyzing network paths of all *retxs* like 007 [9] is also infeasible in our cloud regions. Secondly, raw *retxs* are too 'noisy' to reflect anomalies. Massive applications exhibit distinct patterns and sensitivities concerning *retxs*, due to the different QoS levels, resource quotas and load variations. Internet access is also unstable compared with intranet access. Thirdly, anomaly routing is still challenging with at least five attribution teams (Figure 1) and traffic scenarios (Figure 4) involved.

To resolve the deployment challenges in a cloud-native era, AND substantially goes beyond currently deployed systems with three new designs of anomaly collecting, detecting and diagnosing using the single metric of *retx*.

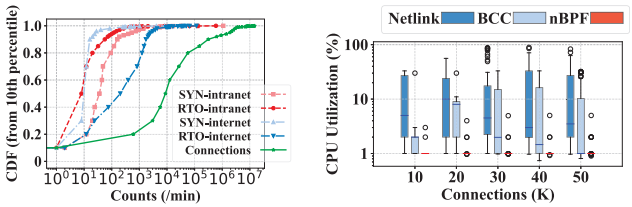
Efficient Anomaly Collecting via nBPF Tool. AND performs statistics and filters on raw *retxs* events via a dedicated nBPF tool, to lower CPU/storage overheads of collecting massive *retxs*. nBPF uses an eBPF-based kernel program to extract prominent anomalies like retransmission timeouts and fails of connection establishment. nBPF leverages user-space filters to record *retxs* counter for anomaly detection and sample *retxs* details (TCP 5-tuples and process info) for anomaly diagnostic. Last but not least, nBPF proposes statistics of *retxs* events per (micro)service/container (source) and per traffic scenarios (destination) to delimit the scope of anomalies at the collecting phase. nBPF filters out $> 90\%$ *retxs* events of 'noises' and also facilitates anomaly detection and routing.

Minute-level Anomaly Detection. AND designs minute-level detection procedure for millions of time series (*retxs* counters of millions of IPs). AND adopts a multi-level time-series clustering to distinguish diverse *retxs* patterns. While partial time series are 'stable' with small 'noises' (41.06%), other time series cannot be processed by simply dropping (low recall) or threshold-based filtering (low accuracy). AND designs lightweight feature engineering, which eliminates 'noises' by prediction and penalizes/scales features of low/high-quality time series. AND adopts a hybrid implementation of offline clustering/predication and online detection for low latency and cost. AND extracts $< 1\%$ anomalies from 'noisy' *retxs* and ensures high detection recall.



(a) #IPs reporting *retxs*. (b) #Total count of *retxs*.

Figure 5: Statistics of *retxs* from a large-scale region. (a) The number of container IPs reporting *retxs* reaches a million level. (b) For intranet access, short flows burst many SYN *retxs* during connection establishment while retransmission timeouts (RTOs) are much less with a relatively stable network. For Internet access, applications usually use long connections and exhibit many RTOs as the network is not stable.



(a) Connections vs. *retxs*. (b) CPU overheads.

Figure 6: Overhead analysis. (a) The distribution of the number of connections and *retxs* at (micro)service levels from a large-scale region. (b) Overheads of collecting metrics: Netlink [7] and BCC [1] collect TCP info of all connections via `AF_NETLINK` and eBPF [2] respectively. nBPF performs filtering and statistics on raw *retxs*. We use a typical virtual machine (VM) instance [3] in public clouds for testing.

Multi-problem-domain Anomaly Diagnostic. AND routes anomaly to multiple attribution teams and traffic scenarios via the single metric of *retx*. The key insight is that anomalies of different problem domains (*e.g.*, networks, end hosts) exhibit distinct distributions of *retxs*, after correlating source/destination (SRC/DST) IPs in *retxs* details with application/network attributes. AND builds a supervised routing model and designs feature sets of anomalies for multiple target teams and scenarios, according to anomalies/failures observed from production. The routing model helps locate anomalies and find root causes by combining with domain-specific monitors and metrics. It also embodies good generalization ability, *e.g.*, extending from a limited scope to all cloud regions.

AND has been deployed in Alibaba cloud for over three years. It processes more than one billion *retxs* events per hour and reduces data volume by orders of magnitudes (§6.2). The routing model achieves $\sim$ 95% recall/accuracy in detecting/routing anomalies to target network teams in the long-term evaluation (§7.2). AND enables minute-level anomaly detecting and routing (§9.1). AND also achieves high coverage of anomalies in the end-to-end path, from networks to end hosts and even abnormal application behaviors, and complements the fade area of existing systems (§9.2).

The contributions of this paper are summarized as follows:

- We reveal issues of impact assessment and anomaly routing in operating large-scale production clouds. To tackle these issues, we introduce AND, a unified application-network diagnosing system exploiting a single metric of TCP *retx*.
- We reveal that production clouds exhibit massive and noisy *retxs* with complex problem attribution and traffic scenarios. AND thus designs the effective collecting tool, detection procedure and diagnostic mechanism to extract, identify and delimit anomalies step-by-step.
- We iteratively deploy and optimize AND in Alibaba cloud. Currently, AND covers all of our cloud regions and millions of (micro)services/containers. We also share experience on usage/scope of AND and lessons learned about Artificial Intelligence for IT Operations (AIOps).

2 Background & Motivation

We introduce the background of cloud architecture and overview of cloud traffic. On this basis, we present the operational experience and also motivation of AND.

2.1 Overview of Cloud Architecture

Figure 1 shows a multi-layer service model of Alibaba cloud, where each layered component is usually attributed to a dedicated team. Software services, such as e-commerce [29], big data [16, 53], databases [13], belong to their respective application teams; platform services including containerized platforms [27, 40], virtual machines (VMs) and their operating systems [14], are in charge of dedicated platform teams; infrastructure services including virtualization and host machines, virtual or physical networking, are handled by the corresponding infrastructure, virtual or physical networking teams [22, 32, 52].

Overview of Cloud Traffic. Cloud applications go through each layer of components in the end-to-end path (Figure 2). As shown in Figure 4, applications run on containers and VMs, and communicate via kernel stack and virtualized netdev [52]. The traffic is then forwarded via vSwitch [35] and physical NIC at host machines or node controllers (NCs). The cloud gateways [32] as the cores of virtual networks perform stateless and stateful network functions, including forwarding gateways among virtual private clouds (VPCs), stateful load balancing (SLB), network address translate (NAT), access control list (ACL). The physical network consists of multi-level switches/links, and the cloud enterprise network (CEN, a dedicated leased line network between cloud regions).

On this basis, we summarize five common scenarios of cloud traffic (Figure 4): ① Cloud services actively access internet data centers (IDCs). ② User clients request services hosted by the cloud, which return responses. ③ Cloud services access other intranet services via SLB. ④ Cloud services access each other directly via forwarding gateways. ⑤ Cloud services access each other cross regions.

2.2 Operational Experience

The complex layered architecture poses great challenges to the operation of cloud services — anomalies may happen at each layer [20,22,57]. The operation teams at each layer have built abundant monitoring and diagnostic tools including end-host monitors [9,10,30,39,43,49], virtual network [15,38,57] or physical network probings/telemetries [11,22,44,47,56], to quickly detect and locate anomalies. Nevertheless, we still lack a unified monitoring/diagnostic system that can assess the impact of anomalies/failures at a (micro)service level, quickly route anomalies to attribution teams, and cover the entire cloud with low overheads. Next, we elaborate on key issues observed from operational experience.

How to estimate the impact of anomalies on real application traffic? The networking teams build large-scale probing systems to monitor virtual/physical networks [22,57]. However, probing systems cannot tell whether application traffic is affected by anomalies and even overlook severe failures. Specifically, they probe each node/link with equal weight, while partial nodes/links carrying critical (micro)services that many applications depend on should have larger weights. Probing results on these critical nodes/links may be covered by noises of network jitters (Figure 3). We observe many cases where failures in critical (micro)services are not perceived by probing systems. Real case studies of SLB and Redis services are presented (Cases A and C in §8). Other monitors on the platform or infrastructure layer face similar problems. Even though anomalies are detected, they cannot tell whether or how many applications are affected.

Whether applications work normally after network alteration or recovery? The network teams often perform network alterations or execute traffic migration for failure recovery. They need to know as soon as possible whether applications work normally. Currently, network teams verify the running status of applications by contacting each application team, which takes a long feedback cycle. During several network alternations, only partial applications' traffic is migrated successfully, resulting in unavailable services and large revenue losses. Similar issues can be avoided if we can find a way to verify the applications' status quickly (Case B in §8).

Experience #1: To conduct impact assessments from application perspectives, the diagnostic system needs to monitor applications' in-band traffic at granularities of connection and (micro)service [17,45].

How to perform fast and accurate anomaly routing to attribution teams? Anomalies are usually caused by the dependent (micro)services or shared network infrastructure with a (micro)service-based architecture [18]. Specific application monitor has limited scope and cannot tell reasons for anomalies directly. However, correlating application metrics with problematic (micro)services and network events is non-trivial, due to the inconsistent data format and semantics (program tracing/logging vs. connection-oriented metrics [20,50]). To this end, application teams report their failures separately and contact related teams respectively, resulting in high communication costs and low efficiency for failure recovery. We present several cases where a unified diagnostic system facilitates anomaly routing and locating (Case D-F in §8).

Experience #2: To improve the efficiency of diagnosing, clouds need a unified diagnostic system to correlate layered components and route anomalies to attribution teams.

How to achieve full coverage with low overheads? The number of container instances or IPs has increased to a million level in Alibaba cloud (Figure 5). The connection's scale on long-tail instances also reaches a million level (Figure 6a). The overheads of monitoring end-to-end anomalies for millions of IPs and billion connections are considerable.

Experience #3: As an always-on service, the monitoring system should be carefully optimized for low computational and storage overheads, and lower the long-term capital/operating expenses (CapEx/OpEx).

Motivation: We begin to build and deploy the application-network diagnosing (AND) system since 2019. AND aims to tackle the above issues in production clouds.

3 Design Rationale & Challenges

3.1 Design Rationale

AND takes TCP Retransmissions (*retxs*) as basic metrics to build unified anomaly monitoring and diagnosing capability covering the whole cloud. The key insight is that *retxs* are effective signs of anomalies, inherently exist in in-band traffic, and correlate applications to fundamental platforms and infrastructure end-to-end. Figure 7 shows the coverage of AND, from a bottom-up perspective, including physical and virtual networks, physical NIC, vSwitch, virtualized netdev, kernel stack and even abnormal application behaviors.

AND is universal for different applications (Go, Java, *etc.*) and has no dependencies on hardware devices (programmable NICs/switches [20, 47]). AND also easily extends to other reliable transports like QUIC [25] and Reliable Connection (RC) of RDMA [24], where *retxs* and timeouts can be used to indicate anomalies. Note that the basic idea of collecting TCP retransmissions and statistics for faulty-link locating and failure diagnosing [9,10,39,49] has been proposed in existing works. AND adopts a similar idea to them from this point.

3.2 Challenges

As the deployment of AND in the complex cloud environment, we still face challenges of anomaly collecting, detecting and diagnosing using *retxs*. AND has continued to evolve in iterative deployment, resulting in fundamental differences with existing works.

First, production clouds unexpectedly exhibit massive *retxs* — the number of *retxs* of single (micro)service instance

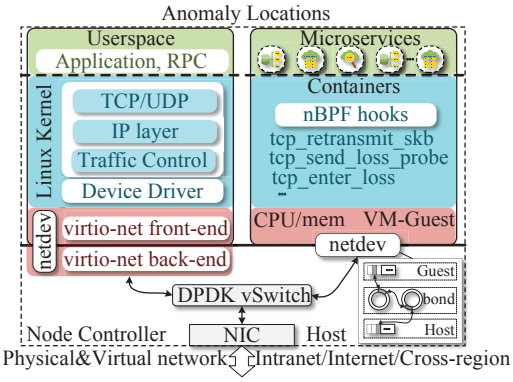


Figure 7: AND's coverage of anomalies.

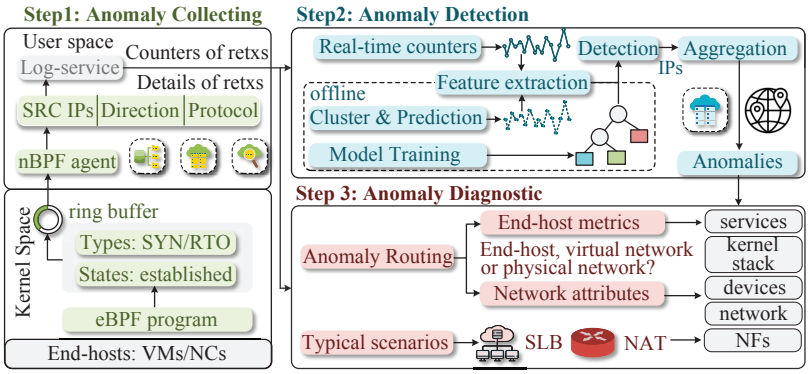


Figure 8: Overview of AND's architecture.

reaches $10 \sim 10^5/min$ and has a long-tail nature (Figure 6a). For each *retx*, 007 [9] tries to find the network link/path that causes packet drops via Traceroute. In our cloud regions, collecting and analyzing each *retx* like 007 easily overloads CPUs of end hosts and switch's control plane.

Challenge #1: The clouds require efficient collecting tools to perform filtering/sampling on raw *retxs* at (micro)service granularity while reserving the critical info of anomalies.

Second, the raw *retxs* are too 'noisy' to reflect anomalies that cause SLA violation. With mixed deployment in public clouds, applications apply computing/network resources according to their QoS requirements and exhibit distinct patterns of *retxs*. For example, latency-sensitive services [8] cannot tolerate timeouts and have few *retxs*, but background tasks [16, 53] have many *retxs* in daily operation, so long as tasks complete on time. These applications also deploy many (micro)service instances across cloud regions, constituting large 'noises'.

Challenge #2: The clouds require real-time anomaly detection to filter out 'noises' and extract anomalies from millions of time series of *retxs*.

Last but not least, anomaly routing and locating are non-trivial with at least five attribution teams (Figure 1) and five traffic scenarios (Figure 4) involved. The pioneering Scout [19] targets incident routing for the physical networking team. NetPoirot [10] adopts failure injection and identifies failure attribution (client, network, and remote server) using TCP statistics at one end host. However, the complex layered dependencies and traffic scenarios hinder the direct usage of these works in a cloud-native environment.

Challenge #3: The clouds require a new diagnostic mechanism to perform fast and accurate anomaly routing among multiple target teams and traffic scenarios.

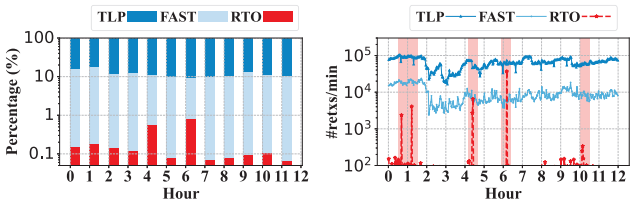
4 Overview

Next, we introduce the design and deployment of AND in practice to resolve the above challenges. Figure 8 shows the overall architecture of AND incorporating anomaly collecting, detection and diagnostic. The process from anomaly collecting to routing takes less than 1 minute.

Anomaly Collecting (Challenge #1). AND monitors *retxs* at end hosts via a dedicated tool, namely nBPF (§5). nBPF designs eBPF-based kernel filters to extract prominent anomalies that impact application performance, *i.e.*, fails of connection establishment and retransmission timeouts. nBPF also devises user-space filters to record accurate *retxs* counters for anomaly detection and sample *retxs* details (TCP 5-tuple and even process info) for anomaly diagnostic. Last but not least, nBPF proposes statistics of *retxs* events per (micro)service (SRC) and per traffic scenarios (DST) to delimit the scope of anomalies in the collecting phase. nBPF effectively filters out $> 90\%$ of *retxs* and achieves low overheads in extreme stress tests with million connections. The filtering rules also ensure a high coverage rate of anomalies and help anomaly routing among multiple scenarios.

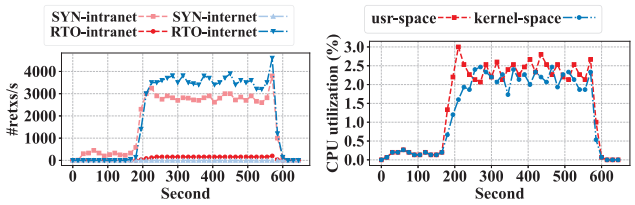
Anomaly Detection (Challenge #2). AND then performs real-time anomaly detection on time series of *retxs* counters (§6). AND adopts multi-level time-series clustering to distinguish distinct *retxs* patterns with respect to frequency, stability, seasonality, *etc.* According to the clustering results, AND designs lightweight feature engineering and extracts normalized features for anomaly detection. The compute-intensive clustering and model training are conducted in the offline phase, while the feature extraction and anomaly detection are executed in real-time. Finally, AND achieves minute-level detection for millions of data streams (time series of millions of IPs $\times$ multiple scenarios). AND extracts $< 1\%$ abnormal IPs from 'noisy' time series of *retxs* and guarantees high recall in anomaly detection.

Anomaly Diagnostic (Challenge #3). The abnormal IPs are aggregated by application/network attributes and exported to the diagnostic process (§7). The key insight is that anomalies in multiple scenarios, *e.g.*, intermediate networks, end hosts, *etc.*, exhibit different distributions of *retxs*, after correlating *retxs* details with application/network attributes. AND builds a supervised anomaly-routing model using single *retxs* metrics and designs feature sets of anomalies for multiple target teams or scenarios. The routing model also expands



(a) Percentage of *retxs* counters. (b) Trend of *retxs* counters.

Figure 9: The noisy *retxs* without filtering (collected from tens of IPs of search service in one day).



(a) The variations of *retxs*. (b) CPU overheads.

Figure 10: The CPU overheads of nBPF in extreme stress tests with million connections.

the training sets and executes re-trains in a self-iterative way. In the long-term evaluation in the production cloud, AND achieves $\sim 95\%$ recall/accuracy in detecting/routing network anomalies. The routing model helps to locate anomalies in specific scenarios and embodies generalization ability to more problem domains.

5 Anomaly Collecting

5.1 Design of nBPF Tool

The nBPF tool consists of a kernel-space eBPF program and a user-space agent (Figure 8). eBPF [31, 46] is universally supported in Linux distributions (from Linux kernel 3.15+), which dominate majority in Alibaba clouds. Note that the idea of collecting *retxs* is general while the implementation may vary according to different OS distributions [9] and even transports [24, 25]. The core of nBPF designs a configurable kernel- and user-space filtering framework with the validated rules to extract prominent anomalies from massive *retxs*.

Kernel-space filtering on *retxs* types. The eBPF program hooks *retxs*-related kernel functions to monitor *retxs* from each (micro)service/container at the same host, as multiple containers with isolated network namespace actually share one monolithic kernel stack [58]. nBPF also distinguishes various types of *retxs*.

In the initial deployment, AND collects all *retxs* without filtering. We observe that applications exhibit massive *retxs*, where TLP (Tail Loss Probe) and FAST *retxs* take the majority (around 99%) but cannot reflect the real anomalies, as shown in Figure 9a. Because the end-host/network jitters easily cause massive TLP or FAST *retxs*, *e.g.*, in mice flows of remote process calls (RPCs) between (micro)services [36, 55]. nBPF thus filters *retxs* by types and connection states in the collecting phase.

First, nBPF monitors SYN *retxs* during connection establishment, which indicates that applications cannot establish service. Second, nBPF only extracts RTO *retxs* for established connections (around 0.1%). As shown in Figure 9b, RTO *retxs* are effective signs of application/network anomalies experiencing long timeouts, since kernel stacks adopt a default timeout of 200*ms*. The practice demonstrates that such filtering rules achieve a good trade-off between collecting overheads and coverage rates of anomalies.

User-space statistics and samplings on *retxs* details. The reported *retxs* events by the eBPF program contain the detailed TCP 5-tuples and even process info. To lower collecting and processing overheads, the user-space agent further performs statistics and samplings on raw *retxs* events/details. On the one hand, nBPF records counters of *retxs* in container-level granularity (SRC) with fixed time intervals for anomaly detection. On the other hand, *retxs* details are indispensable for anomaly diagnosis. However, massive *retxs* events/details incur large overheads for uploading and processing (Figure 6a). nBPF thus performs samplings on raw *retxs* events.

Last but not least, nBPF distinguishes multiple traffic directions and scenarios via configurable filters on DST IPs (Figure 4), to coarsely delimit the scope of anomalies when performing statistics of *retxs* counter. As *retxs* counters are lightweight statistics, nBPF supports finer-grained statistics of *retxs* with respect to application/network attributes. Finally, the *retxs* counters and details are uploaded for further anomaly detection and diagnosis.

5.2 Deployment & Evaluation

Performance and Stability Consideration. The initial version of nBPF is implemented with Go and then reconstructed with Rust, to process *retxs* events more efficiently. At last, nBPF tool is implemented with 5000 lines of Rust code (nBPF agent) and 1000 lines of C code (eBPF program). nBPF adopts a time interval of 15*s* for statistics of *retxs* counters, which is fine-grained enough to capture anomalies. nBPF samples the first 23 *retxs* for details, which is the $90th$ percentile of *retxs* counters in intranet scenarios.

The CPU/memory usage of nBPF is carefully optimized to avoid interference with applications [42, 48]. The nBPF agents read *retxs* events from kernel space via zero-copy ring buffer. The CPU quota is limited to 5% of one core and the memory quota is limited to 30MB via `cgroup`. For stability consideration, nBPF tools are automatically deployed according to the IP list (VMs/NCs) with multi-phase canary testing. nBPF agent also reports empty *retxs* counter as the heartbeat.

Coverage Analysis. AND deploys nBPF at guest OS or bare-metal servers. The coverage of AND thus depends on how many VMs/NCs are deployed. The nBPF tool relies on eBPF

features of Linux kernel, without requiring any other specific hardware/software. The early-version kernels cannot support eBPF and only occupy the minority of machines (18.43%). nBPF adopts `kprobes` and `perf_events` to monitor *retxs* effectively, which are introduced from kernel 4.9+ [2]. AND covers 96.91% of the rest of machines with kernel version supported (81.57%). The rest machines (3.09%) are managed by another corporation and are thus not covered.

As shown in Figure 5, the number of container IPs reporting *retxs* in one hour reaches an order of 10^6. AND also meets our design goal of covering Alibaba proprietary business, including computing platform [4,6], database, mobile shopping, *etc.* Besides, the kernel version keeps upgrading and the old kernels/machines will expire gradually. The coverage of AND thus continues to increase.

Overhead Analysis. AND provides always-on service. Here we focus on the CPU overheads of nBPF. The long-term OpEx for data processing is discussed in §6.2.

CPU Overheads. As shown in Figure 6b, we compare the CPU overheads of nBPF with end-host monitoring tools. Existing works [39,43,49] collect connection-/packet-level metrics at end hosts. For example, they adopt Netlink [7] to collect TCP info from the kernel. The burst loads easily cause a full occupation of a single CPU core. The BCC [1] tools monitor TCP connections using eBPF and reduce the overheads compared to Netlink, however, the peak CPU utilization still exceeds 10%. As a comparison, the CPU overheads of nBPF keep around or below 1%, because nBPF only collects *retxs* events with dedicated optimizations like filtering/sampling, zero-copy collecting, *etc.*

Stress Tests. We also deploy nBPF agent at a proxy server and the connections of the high-frequency trading services reach a million level in stress tests. As shown in Figure 10, system loads of both clients and real servers increase slightly during $0 \sim 200s$ and the average CPU utilization of nBPF keeps around 0.3%. Interestingly, the proxy server bursts massive SYN *retxs* to real servers and RTO *retxs* to clients (Figure 10a), because clients already build connections with proxy servers while proxy servers try to access real servers. The *retxs* counters also burst after many connections break when stress tests are finished (around 550*s*). nBPF limits the CPU usage in such extreme cases and the CPU loads thus keep below 5% (Figure 10b).

6 Anomaly Detection

6.1 Design of Detection Procedure

The clouds host massive applications with mixed deployments. These applications exhibit distinct *retxs* patterns and sensitivities to *retxs*, with different QoS levels, resource quotas and load variations. Even though AND performs filtering in the collecting phase (§5), there still exists large 'noises' in the time series of *retxs* counters from millions of IPs. On the one hand, dropping 'noisy' time series will lose valid data and lower coverage, while using 'noisy' data directly will affect accuracy. On the other hand, the simple threshold- or rule-based methods cannot be applied to various *retxs* patterns.

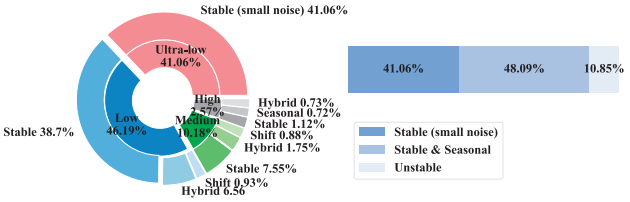


Figure 11: Clustering of time series (intranet): (1) Stable time series with small 'noise' take 41.06%; (2) Stable and seasonal time series with predictable 'noise' take 48.09%; (2) Unstable time series (mean-shift or hybrid) take 10.85%.

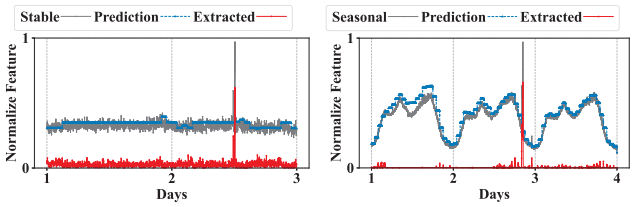
To this end, the core idea of the detection procedure adopts a multi-level clustering to distinguish *retxs* patterns and then designs lightweight feature engineering to extract effective features for anomaly detection.

Multi-level Clustering. The algorithm performs a two-level time-series clustering to capture different 'noisy' patterns. The time series are first clustered according to the occurrence frequency of *retx*. Specifically, time series are divided into four levels from low to high frequency to intuitively reflect the 'noise' level. On this basis, AND further classifies time series into stable type and unstable type (seasonal/trend, mean-shift and hybrid types), according to the stability, seasonality and distribution of windowed means. Figure 11 presents the distribution of *retxs* patterns in cloud intranet. Most time series are stable or seasonal as the network environment of the intranet is relatively stable. A large portion of time series still exhibit 'noisy' patterns and cannot be filtered simply using thresholds.

Feature Engineering & Detection. According to clustering results, the algorithm designs feature engineering for filtering and denoising. First, the feature values are obtained by differencing real-time counters with prediction values of 'noises'. Second, the feature values are penalized and scaled according to the stability and seasonality of the time series. For example, time series that exhibit unstable/unpredictable 'noise' can hardly extract true anomalies and are prone to be penalized. The extracted features either approximate stable time series or are penalized to small weights. The featured values are also normalized and can be applied to detection directly, using a pre-trained model like isolation forest (IF) [28].

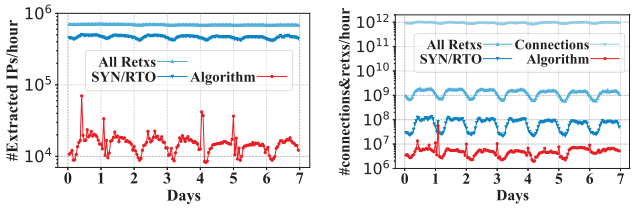
6.2 Deployment & Evaluation

Offline & Online Processing. Real-time is the key requirement of the detection algorithm for fast anomaly/failure discovery and recovery. To achieve this goal, AND adopts a hybrid offline and online processing via the low-cost MaxCompute [6] and real-time Flink [4] respectively. The offline stage performs compute-intensive clustering, prediction and



(a) Stable time series. (b) Seasonal time series.

Figure 12: Effects of detection procedure.



(a) Extracted IPs. (b) Connections vs. *retxs*.

Figure 13: AND only extracts effective anomalies.

model training, while the online stage only extracts real-time features and then detects anomalies.

Offline Clustering. AND uses offline clustering for 'day+1' detection. Partial container IPs may occasionally be assigned to other applications and have different *retxs* patterns compared with the previous clustering results. The time series of these IPs will be identified and penalized in feature engineering. The clustering results will be updated the next day.

Offline Prediction. AND designs dedicated offline prediction for different clustering of time series (Figure 11). On the one hand, partial time series have an ultra-low frequency of *retxs*. AND adopts a simple difference with constants as features. On the other hand, AND adopts a lightweight prediction to eliminate the 'noise' of time series, based on the expectation and variance of history windows.

Offline Training & Online Detection. For offline training, AND randomly samples time series from a one-month period and also includes time series of anomalies/failures to enhance the robustness. The real-time features adopt simple arithmetic calculations using prediction values and penalizing/scaling factors pre-processed offline. Finally, the online detection takes real-time features as inputs and outputs the abnormal IPs and timestamps for further aggregation and diagnosis.

Effective Data Filtering. Next, we demonstrate that the detection procedure effectively extracts features of anomalies and also reduces data volume.

As shown in Figure 12, we intuitively show the effects of the detection procedure, taking stable and seasonal time series as examples. The detection procedure eliminates the 'noise' of time series by lightweight prediction. The features are also penalized and scaled via coefficients in cyclic windows and variance in short-time windows.

Data Volume in Daily Operating. We use data volume to intuitively reflect the computing/storage overheads in daily operations. Figure 13 presents the number of collected items (IPs/*retxs*/connections per hour) from a large-scale region in one week. The number of IPs reporting all *retxs* or SYN/RTO *retxs* approach an order of 10^6. AND further identifies the true anomalies and reduces the reported IPs to an order of 10^4 (Figure 13a). The connection-level monitors need to collect one trillion (10^{12}) connections per hour at the region level. As a comparison, the number of items in all *retxs* details achieves 10^9. The filtering rules and anomaly-detection algorithms further reduce this number to an order of 10^7 and 10^6 respectively (Figure 13b). AND only extracts *retxs* details (from MaxCompute) for detected anomalies. Both filtering rules and detection procedures are effective in reducing computing/storage overheads and identifying true anomalies.

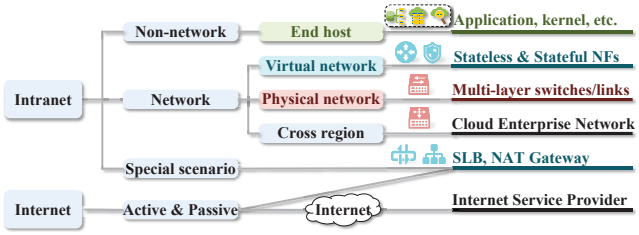


Figure 14: Anomaly Diagnostic.

7 Anomaly Diagnostic

7.1 Design of Diagnostic Mechanism

As shown in Figure 14, AND uses a hierarchical diagnostic mechanism to narrow the scope of problems step-by-step. First, AND distinguishes anomalies between intranet and internet traffic in the collecting phase. AND thus avoids interference from the unstable network environments of the internet in anomaly routing. After that, AND routes anomalies among networks, non-networks (end hosts, applications, *etc.*) and special scenarios like SLB/NAT/Internet. Finally, AND analyzes the location and root cause of anomalies by correlating domain-specific metrics and monitors.

Anomaly Routing Model. The routing model exploits the distribution of *retxs*, *i.e.*, SRC/DST IPs in *retx* details aggregating by application/network attributes, as features to classify network and non-network anomalies. AND adopts a classification model based on the gradient boost decision tree and collects labeled anomalies/failures in practical operations as training sets. The main challenges lie in determining feature sets considering various anomalies in multiple scenarios. We empirically choose and optimize the feature sets according to observations of anomalies from production environments.

Experience on Feature Sets. AND correlates *retx* details with application and network attributes and builds access graphs from multiple dimensions as feature sets of the routing model.

For end-host attributes, AND considers IP-level convergence and aggregates container IPs by VMs/NCs. The *retx* details exhibit IP-level convergence to a small number of

hosts (VMs/NCs) due to the end-host anomalies, *e.g.*, the top several hosts take the majority of total *retxs*.

For application attributes, AND constructs access graphs among attribution teams of applications. The large proportion of core applications as source/destination usually reflect their own problems. Small-scale anomalies between instances of the single application are pruned and processed separately.

For network attributes, we construct access graphs at the region and available zone (AZ) levels. We also consider the multi-layer topology of physical and virtual networks for further anomaly routing.

Last but not least, AND directly correlates *retx* details with special scenarios like SLB/NAT/Internet, via virtual IPs (VIPs) or sessions at SLB/NAT gateways. For example, clients and real servers are aware of VIPs (with SLBs or NCs performing NAT [33]), while internet traffic is served by fixed SLB/NAT gateway clusters as inlet/outlet at the region level.

Anomaly Locating and Root Cause. AND further provides guidance for anomaly locations based on the routing results.

End host & Application. For end-host and application anomalies, AND identifies the problematic SRC/DST IPs of (micro)services/VMs/NCs. AND further correlates end-host metrics (CPU/memory utilization, ingress/egress traffic and packet drops at kernel-stack/netdev/vSwitch/NIC) and application-specific monitors, to find the root cause (Case D).

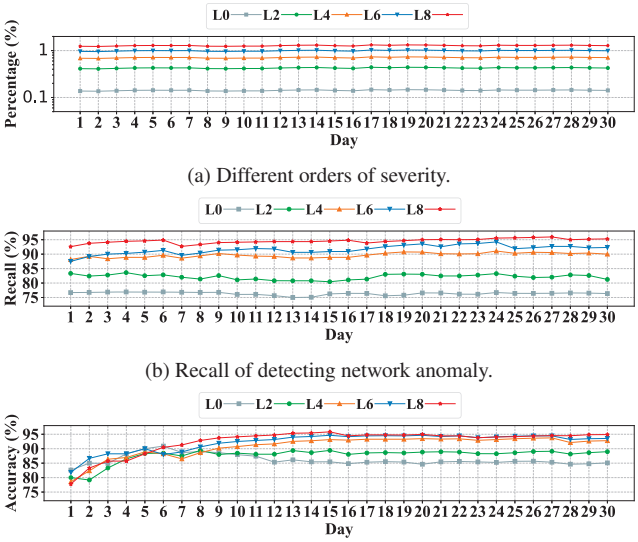
Physical & Virtual Network. For network anomalies, AND determines the scope of anomalies at the region and AZ levels. For example, inter-region anomalies are usually caused by the unstable CEN (§8.2). AND also correlates *retx* details with network topologies/attributes, *e.g.*, multi-layer ToR/aggregation/core switches in physical networks, and vSwitches/forwarding-gateways/VPCs in virtual networks. The abnormal hosts will aggregate to physical- or virtual-network failures and even tell which tier a faulty device/link is located in.

SLB & NAT Gateway. AND identifies gateway anomalies at the instance or cluster level. At the instance level, *retx* details converge to VIPs for source/destination NAT, which are usually related to specific applications. At the cluster level, *retx* details are distributed among multiple nodes or the whole cluster and many applications are affected. AND further checks loads of applications and SLB/NAT clusters and other related metrics to find the root cause (Case A).

Internet. Internet traffic experiences both intranet and internet paths, including SLB/NAT clusters, backbone routers, internet service providers (ISPs), *etc.* After excluding the cloud vendors' problems (SLB/NAT cluster or backbone router), AND routes anomalies to the internet direction (Case F).

7.2 Deployment & Evaluation

Iterative Model Training. AND iteratively expands training sets and executes re-train on a daily basis. The routing model



(a) Different orders of severity.

(b) Recall of detecting network anomaly.

(c) Accuracy of routing network anomaly.

Figure 15: Recall & Accuracy of AND in production.

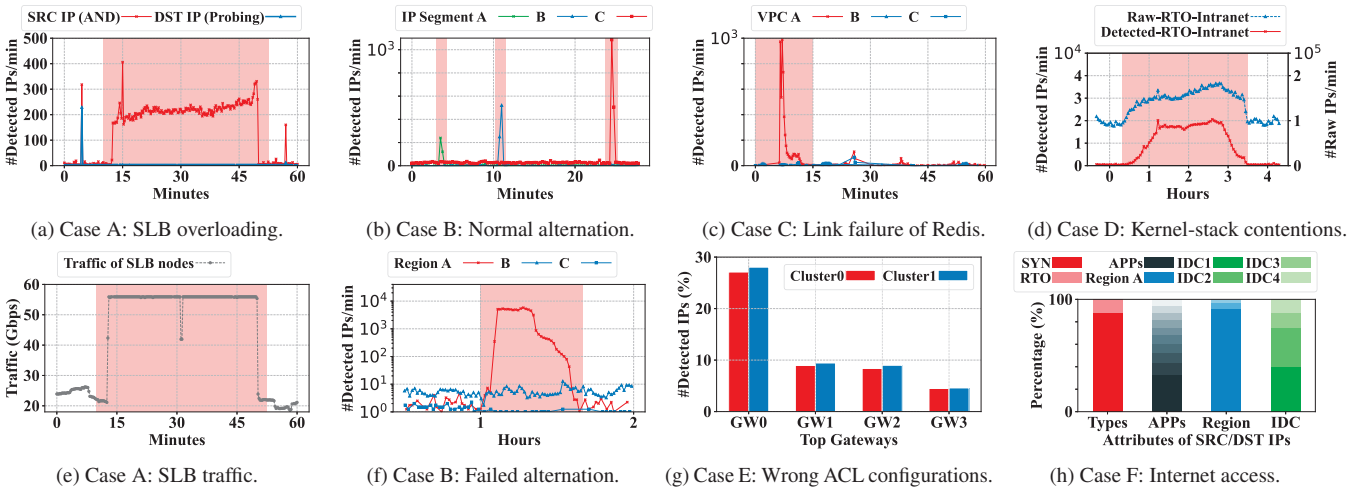
labels each detected anomaly. Initially, the routing model only ensures correctness for partial scenarios with abundant cases. The operators need to verify labels that have a low confidence level. The accuracy of AND gradually improves with the extended data sets. AND can automatically expand data sets with no human involvement as a high confidence level has been achieved.

Evaluation in Production. We evaluate the capability of AND in anomaly detecting and routing in production deployment. We conduct a long-term evaluation by comparing detecting/routing results of AND with other diagnostic tools in daily operations (§8.2).

Methodology. We focus on network anomalies to evaluate the routing model. On the one hand, the clouds have built mature probing and telemetry systems for physical and virtual networks [15, 22, 56, 57], which can be used as baselines for evaluation. On the other hand, delimiting network anomalies is a challenging task with many application or end-host anomalies as interference. We also validate other types of anomalies in daily operations by inspecting domain-specific monitors.

To capture the sensitivity of AND to anomalies, we define ten orders of severity according to the scope of anomalies. We only consider anomalies that exceed a defined severity, *i.e.*, the number of abnormal IPs detected by AND exceeds 1/1000 (L0) to 1/100 (L9) of the total number of monitored (micro)services/IPs. Given a specific severity, recall is defined as the percentage of network anomalies routed by AND among all network anomalies, while accuracy is defined as the percentage of true network anomalies among network anomalies routed by AND.

Recall & Accuracy. Figure 15a shows the percentage of affected (micro)services/containers with different orders of severity, from 0.1% to 1%. Figure 15 also presents the accu-



(a) Case A: SLB overloading. (b) Case B: Normal alternation. (c) Case C: Link failure of Redis. (d) Case D: Kernel-stack contentions.

(e) Case A: SLB traffic. (f) Case B: Failed alternation. (g) Case E: Wrong ACL configurations. (h) Case F: Internet access.

Figure 16: The abnormal IPs and the aggregations of SRC/DST IPs (*retxs* details) facilitate failure detection and routing.

mulative recall and accuracy since the initial deployment of the routing model in a one-month period. AND achieves a relatively stable recall in detecting network anomalies, from 75% to 95.96% as shown in Figure 15b. The accuracy of routing network anomalies improves gradually with the expanded data sets. In the stable phase, AND achieves 84.61% to 95.83% accuracy in routing network anomalies, considering different orders of severity (Figure 15c).

Analysis. We then analyze the false positives and false negatives of AND considering network anomalies. The false positives are mainly attributed to the wrong routing conclusions. For example, end-host or application anomalies may be wrongly routed to network anomalies (§9.2). Note that network failures may not be covered by probing systems in particular cases (§8). We also consider these cases which are confirmed as network failures by operation teams.

The false negatives are due to the different design rationales of AND and probing systems. AND monitors in-band traffic of applications and reports anomalies only if application traffic is affected. On the other hand, probing systems adopt probing traffic and target anomalies from the network perspective. For example, probing systems are sensitive to network jitters (slight packet drops of network links), however, application traffic may not pass the problematic links or applications have no network I/O when anomalies happen.

In summary, both recall and accuracy of AND gradually increase with more severe network anomalies. The impacts on application traffic also increase with the larger scope of anomalies. While the application may not be affected at L0, there is a high probability that applications are affected at L9.

8 Case Studies

8.1 Impact Assessment & Coverage Analysis

Absence of Impact Assessment. The most widely deployed network probing systems [22,57] lack impact assessment from application perspectives. They cannot tell whether application traffic is affected by anomalies and even overlook large-scale failures.

Case A: SLB overloading. AND reports the number of affected (micro)services (SRC IPs) in anomalies caused by SLB overloading (Figure 16a), where the *retxs* details converge to VIPs served by SLB nodes. As a comparison, the probing systems [22,57] may detect delays/losses when probing paths of SLB nodes (DST IPs), however, cannot tell the accurate scope of impact from the service perspective. After checking the traffic of SLB nodes, we observe that SLB nodes are overloaded and thus drop many packets (Figure 16e). The services recover to normal after operators extend the quota/capacity of SLB nodes.

Case B: Failures in gateway alternations. AND tells whether network alternations affect applications and whether applications recover to normal after emergency actions. For example, network teams often perform gateway alternations/upgrades. Generally, the related IP segments will burst many *retxs* due to connection breaks and then recover to normal (Figure 16b). AND helps detect unexpected gateway-cluster failures quickly during alternations (Figure 16f). The operators then execute roll-back actions and applications recover to normal rapidly.

Case C: Link failures accessing Redis. AND detects a large-scale failure of the instant take-out ordering services (Figure 16c). The ordering services further rely on the in-memory cache services (*e.g.*, Redis [8]) in dedicated VPC and appear many *retxs* to the target VPC. Because the IP-table (8-bit index) is overflowed in the forwarding gateway — the default DST IP (index 0) for return packets is replaced with a wrong value — all requests to Redis receive no responses (forwarded to the wrong DST IP). The network monitors [22, 57] may detect small-scale failures among all network paths but regard them as not urgent. AND also helps to find the real culprits, *i.e.*, failing to access the dependent Redis services.

Limitations in Coverage. The probing systems fall short in a number of scenarios, *e.g.*, kernel stacks, ACL rules and internet traffic. We also demonstrate that a unified diagnostic system like AND facilitates anomaly routing and fault locating.

Case D: Kernel-stack contentions. AND detects a large-scale failure caused by kernel-stack contentions and helps to pinpoint the problematic VMs/NCs (Figure 16d). The kernel plugins are deployed in many hosts to trace the CPU scheduling events of the densely deployed microservices [45]. The frequency of tracing changes from 5*s* to 1*s* in one release, and causes severe contentions with `ksoftirqd` of packet processing [5, 12], further resulting in many *retxs*. The intermittent probing systems deployed at NCs may not perceive such kernel-stack anomalies inside VMs (§11).

Case E: ACL problems. The out-band probings may fail to capture problems experienced by in-band traffic, *e.g.*, ACL rules will allow probing packets by default. As an implementation of service-level access control, the sidecar (proxy in service mesh [34]) injects/verifies identifications as the payloads of SYN packets via TCP fast open (TFO) option. AND detects a large-scale service failure with many SYN *retxs* correlating with gateway clusters (Figure 16g). Because sidecars adopt the wrong ACL configurations for traffic passing NAT gateways. The gateways drop these TFO packets by default to avoid denial of service (DoS) attacks. The services recover to normal after ACL rules are corrected.

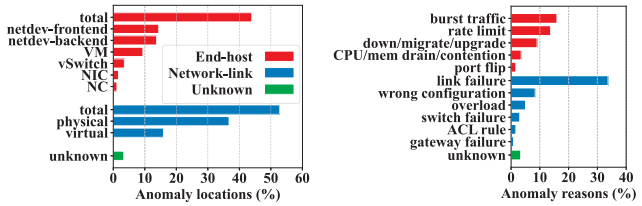
Case F: Internet problems. AND detects service anomalies caused by backbone routers or ISP problems. As shown in Figure 16h, AND reports massive SYN *retxs* from region A to internet direction and many applications are affected (① in Figure 4). AND routes anomalies to the internet outlet of region A (NAT clusters — backbone routers — ISPs). After correlating with syslogs of devices, we finally pinpoint port-down problems at the backbone router. AND also detects the VIP-level anomaly where user clients with common ISP attributes are affected (② in Figure 4), because the ISP wrongly blocks VIP of services.

8.2 Distribution of Anomalies

We analyze the distribution of anomalies/failures and the corresponding root causes, uncovered by AND in a one-month period, as shown in Figure 17.

Anomaly Locations. Figure 17a shows the distributions of anomaly locations. For end-host anomalies, the `netdev-frontend` includes layers above it (guest OS, kernel stacks, applications in Figure 7) and has the largest occupation (14%). For network anomalies, most of the problems happen at physical networks (37%) and the rest problems are due to virtual networks (16%).

Root Causes. Figure 17b shows the root causes of the above anomalies/failures. Most of the end-host anomalies are caused by burst traffic (16%). For example, the `kernel stack` fails



(a) Locations of anomalies. (b) Root causes of anomalies.

Figure 17: Distributions of anomalies.

to process packets and causes drops at `netdev-frontend`. The VMs/vSwitches may also be overloaded by burst traffic exceeding capacity. The rate limit (13%) works at `netdev-backend` and causes packet drops when quotas are exhausted. The down/migrate/upgrade behaviors (9%) of critical modules, *e.g.*, services (containers), VMs, vSwitches, NC, *etc.*, break connections and causes many *retxs*. The CPU/memory drain usually happens at VMs and the physical NIC problems are usually caused by port flips. The CPU contention at the kernel stack leads to a large-scale failure (Figure 16d).

Most of the network anomalies are caused by link failures (34%) or switch failures (3%) of physical networks. The CEN or leased-line failures (19%) occupy the most of link failures for cross-region communications. The virtual-network failures are usually caused by wrong configurations, overloaded NFs, ACL rules and faulty gateways. Besides, there still exist 4% unknowns anomalies that cannot be well explained. For example, the container IPs may already be migrated in function as a service (FaaS) [17] before AND finds the root cause by correlating with other diagnostics tools.

9 Experience

In this section, we introduce experience about how AND is used. We also analyze the scope and limitations of AND in practical deployment.

9.1 Practical Usage

Network Teams. The virtual or physical networking teams detect network anomalies via active probing [22, 57] and in-network telemetry [56], and often perform network alteration or traffic migration. They use AND to verify whether applications are affected and then perform emergency recovery. For example, applications will burst *retxs* and then recover to normal in a successful gateway alteration (Figure 16b). In one unexpected gateway failure during alternation, AND detects persistent *retxs* of the affected applications (Figure 16f). The operators then perform rollback immediately and applications recover to normal.

Application Teams. The application teams observe abnormal metrics via their specific monitors. The operators then query AND to verify *retxs* details of related (micro)services.

If AND reports no anomalies, problems are usually blamed on application-layer logic. Otherwise, AND provides routing results to attribution teams. By correlating *retxs* details with domain-specific monitors, operation teams quickly determine the root causes of the anomaly. AND enables minute-level anomaly detection and routing, and thus improves the efficiency of diagnosing significantly.

Active Alerting. AND also pushes alerts to related teams actively. PaaS and IaaS teams cross-validate the reported anomalies from AND and give feedback on root causes. More importantly, AND complements peculiar anomalies that are not detected by existing monitoring systems (§8). Application teams give feedback on whether QoS is affected by *retxs* anomalies. For example, Redis [8] are sensitive to timeouts, while MaxCompute [53] focuses on the expected deadline of background tasks instead of request-level timeouts.

9.2 Scope & Limitations

Scope & Advantages of AND. AND mainly targets PaaS and IaaS layers since *retxs* work below application layer (Figures 1 and 7) — application-layer anomalies like RPC timeouts may not necessarily trigger transport-layer *retxs* — the kernel stack may return ACKs as usual. Interestingly, AND also detects many anomalies due to abnormal application behaviors including CPU/memory exhausting and unattended service upgrade/migration (§8.2). Application teams have built plenty of dedicated tools to monitor application-layer logic [8, 20, 37, 54]. The key challenge is to correlate applications to networks and tell whether application anomalies are blamed on network failures or whether network malfunctions/alterations have severe impacts on applications.

AND has superiorities in coverage and impact assessment, and thus complements the blind points of currently deployed monitoring systems. First, AND monitors in-band traffic while the probing systems [22, 57] may not perceive anomalies experienced by application traffic. Second, AND monitors *retxs* at (micro)service and connection level for impact assessment from the application perspective.

Anomaly Routing and Locating. AND may occasionally make wrong routing conclusions. For example, end-host anomalies are routed to network teams — failure caused by kernel-stack contention affects many applications/IPs and manifests a similar distribution of *retxs* as network anomalies (Case D in §8). Even though AND gives a wrong delimiting conclusion, it detects the large-scale failure at once and provides details of the affected (micro)services and SRC/DST IPs. The operation teams then use this info as input for fast anomaly diagnosis. Above all, AND identifies the abnormal SRC/DST IPs and coarsely delimits the scope of anomalies. After that, the details of anomalies may be routed to dedicated systems [22, 56, 57], to find the exact locations and root causes.

10 Lessons Learned

In this section, we present lessons learned about unified operating entrance and exploration of AIOps.

10.1 Unified Diagnostic Platform

Our clouds have built mature monitoring systems with clear labor of division. Existing monitors focus on their target domain, *e.g.*, PingSys [22] targets physical networks and Zoonet [57] targets virtual networks. When applications encounter problems, operators check each system and try to correlate anomalies with network events. With so many systems and metrics, the whole process is inefficient in finding the root cause. A unified diagnostic platform facilitates this process, which integrates multiple scenarios like physical and virtual networks and performs fast anomaly routing to target teams. AND enables minute-level anomaly detection and routing.

10.2 Roads toward AIOps

Operating large-scale clouds incurs large OpEx, especially human costs. Introducing AIOps helps to release heavy workloads for human beings, however, still face challenges in practice. There exist huge gaps between available training sets and complex machine-learning models. To employ more features as inputs and more complex models, we need sufficient cases of real anomalies as training sets. AND is a good exploration of AIOps. AND acquires training sets and extracts features of anomalies in a limited scope, *e.g.*, sampling from partial applications and cloud regions. AND then trains the routing model that can be generalized to the whole cloud. AND also iteratively expands training sets, by automatically labeling anomalies with the trained model.

AND combined with AIOps have advantages of scalability and versatility, *e.g.*, extending to more cloud regions and supporting IPv6 protocols, compared with traditional probing systems. With the deployment of more cloud regions, the number of containers/VMs increases exponentially. Full-mesh probing faces telemetry complexity issues and huge overheads [22, 57]. Probing systems only cover partial virtual or physical machines with sampling and path pruning. AND already has full deployment in our clouds and easily extends to more cloud regions, by monitoring application traffic and only collecting anomalies with low overheads. In the early phase, AIOps relies on existing monitoring systems for label and cross-validation. The routing and diagnosing models are expected to achieve better generalization ability with iterative training and optimization.

11 Related Works

Active Probing. Pingmesh [22] targets physical networks in large data centers. VNET Pingmesh [38] and VTrace [15]

Diagnostic Tool Category	Coverage: Probing, Application Traffic	Perspective: Application, Network	Overhead: End-host, Switch, Storage	Deployability: Scope, Limitation
Active Probing: Pingmesh [22], NetBouncer [44], Zoonet [57]	Probing traffic, Intranet	Physical & virtual network	Low, Low, Low	Large data centers and clouds, network only
End-host Monitoring: SNAP [49], PathDump [43], 007 [9]	Application traffic, Intranet	Application & physical network	High, Low, High	Large data centers, high CapEx and OpEx
In-network Telemetry: PINT [11], NetSeer [56], SpiderMon [47]	Application traffic, Intranet	Application & physical network	Low, High, Low	Partial deployment, programmable hardware
AND	Application traffic, Intranet & internet	Application & physical, virtual network	Low, Low, Low	Million (micro)services, cloud-native supported

Table 1: Coverage, perspective, overhead and deployment analysis of existing diagnostic tools and AND.

extend the coverage to virtual networks in the clouds. Zoonet [57] further extends the scope to end hosts, which covers anomalies of vSwitches/NICs and VMs/netdevs via ARP ping. However, the out-band probing may not cover problems of actual service traffic. For example, ACL rules allow probing packets by default and the ARP packets experience different paths with TCP/IP packets of services at the kernel stack. They target non-transient network failures (no shorter than the probing interval [44]) and intranet traffic (internet is uncontrollable [57]). Last but not least, AND reveals for the first time that the probing systems fall short in assessing the impacts of network anomalies on real application traffic in a cloud-native environment.

End-host Monitoring. To correlate applications with network paths end-to-end, existing works collect connection-, link- and even packet-level metrics at end hosts. PathDump [43] and Facebook [39] propose to correlate connections with the network paths by marking packets at each hop and then parsing packets at end hosts. 007 [9] also collects *retxs* and queries the network path of each *retx* via Traceroute. However, Traceroute incurs too much overheads to the switch's control plane with many *retxs*. NetPoirot [10] identifies root causes of failures only using TCP statistics at one host, which relies on artificial failure injections. In all, collecting fine-grained metrics helps a lot in anomaly diagnostics, however, incurs considerable overheads in long-term operation and should be enabled on-demand.

In-network Telemetry. The programmable data plane and in-network telemetry promote novel monitoring systems at switches/NICs [26, 41]. PINT [11] and NetSeer [56] record network-wide statistics and abnormal events at programmable switches respectively. SpiderMon [47] builds a closed-loop between monitoring and posterior diagnosis to achieve low overhead and high coverage. BufScope [20] monitors request-level anomalies of application RPCs by correlating requests at end hosts (SmartNICs) to network paths (programmable switches). While BufScope [20] has been deployed in Alibaba's production storage application, these solutions rely on new hardware (*e.g.*, programmable switches).

Table 1 summarizes the comparisons of AND with existing works. Different from the probing-based systems, AND monitors the actual application traffic and identifies anomalies from the application perspective. Compared with existing end-host monitors, AND extracts anomalies via the single metric of *retxs*, and achieves lower CPU and storage overhead. AND is also readily deployable in the multi-tenant clouds without relying on specific hardware. However, AND is not omnipotent and should be used cooperatively with other tools. For example, The application- or RPC-level monitors [20, 54] complement AND to detect application-layer anomalies. AND also resorts to existing diagnostic tools [22, 30, 57] to locate the exact locations of anomalies.

12 Conclusion

In this paper, we introduce experience in designing, deploying and operating the application-network diagnosing (AND) system in Alibaba cloud. AND exploits a single metric of TCP *retxs* to build the unified monitoring and diagnosing capability, which enables minute-level anomaly detection and facilitates fast failure recovery. According to the operational experience of over three years, AND demonstrates its superiorities from several aspects, including impact assessment at (micro)service levels, multiple-problem-domain anomaly routing, extremely low overheads and generalization ability.

13 Acknowledgment

We sincerely thank the anonymous reviewers for their insightful comments and feedback. This work was supported in part by NSFC grant 62141220, 61972253, U1908212, 72061127001, 62172276, 61972254, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, Alibaba Innovative Research (AIR) Program. Corresponding authors: Linghe Kong (linghe.kong@sjtu.edu.cn) and Xinlei Kang (xinlei.kang@alibaba-inc.com).

References

[1] BPF Compiler Collection (BCC). https://github.com/iovisor/bcc, June. 2023.

[2] eBPF Features by Linux Kernel Version. https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md, June. 2023.

[3] Elastic compute service (ecs). https://www.alibabacloud.com/product/ecs, June. 2023.

[4] Flink - realtime compute for apache flink. https://www.alibabacloud.com/product/realtime-compute, June. 2023.

[5] ksoftirqd - softirq daemon. https://man.cx/ksoftirqd(9), June. 2023.

[6] Maxcompute - conduct large-scale data warehousing with maxcompute. https://www.alibabacloud.com/product/maxcompute, June. 2023.

[7] Netlink - socket-based communication between user and kernel processes. https://man7.org/linux/man-pages/man7/netlink.7.html, June. 2023.

[8] Redis. https://redis.io/, June. 2023.

[9] 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 *USENIX NSDI*, 2018.

[10] Behnaz Arzani, Selim Ciraci, Boon Thau Loo, Assaf Schuster, and Geoff Outhred. Taking the blame game out of data centers operations with netpoirot. In *ACM SIGCOMM*, 2016.

[11] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. PINT: Probabilistic in-band network telemetry. In *ACM SIGCOMM*, 2020.

[12] Qizhe Cai, Shubham Chaudhary, Midhul Vuppalapati, Jaehyun Hwang, and Rachit Agarwal. Understanding host network stack overheads. In *ACM SIGCOMM*, 2021.

[13] Wei Cao, Yingqiang Zhang, Xinjun Yang, Feifei Li, Sheng Wang, Qingda Hu, Xuntao Cheng, Zongzhi Chen, Zhenjun Liu, Jing Fang, et al. Polardb serverless: A cloud native database for disaggregated data centers. In *ACM SIGMOD*, 2021.

[14] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *SOSP*, 2017.

[15] 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 *ACM SIGCOMM*, 2020.

[16] Yihui Feng, Zhi Liu, Yunjian Zhao, Tatiana Jin, Yidi Wu, Yang Zhang, James Cheng, Chao Li, and Tao Guan. Scaling large production clusters with partitioned synchronization. In *USENIX ATC*, 2021.

[17] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In *ASPLOS*, 2019.

[18] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *ASPLOS*, 2019.

[19] Jiaqi Gao, Nofel Yaseen, Robert MacDavid, Felipe Vieira Frujeri, Vincent Liu, Ricardo Bianchini, Ramaswamy Aditya, Xiaohang Wang, Henry Lee, David Maltz, et al. Scouts: Improving the diagnosis process through domain-customized incident routing. In *ACM SIGCOMM*, 2020.

[20] Kaihui Gao, Chen Sun, Shuai Wang, Dan Li, Yu Zhou, Hongqiang Harry Liu, Lingjun Zhu, and Ming Zhang. Buffer-based end-to-end request event monitoring in the cloud. In *USENIX NSDI*, 2022.

[21] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. When cloud storage meets {RDMA}. In *USENIX NSDI*, 2021.

[22] 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 *ACM SIGCOMM*, 2015.

[23] Roni Haecki, Radhika Niranjan Mysore, Lalith Suresh, Gerd Zellweger, Bo Gan, Timothy Merrifield, Sujata Banerjee, and Timothy Roscoe. How to diagnose nanosecond network latencies in rich end-host stacks. In *USENIX NSDI*, 2022.

[24] Xinhao Kong, Yibo Zhu, Huaping Zhou, Zhuo Jiang, Jianxi Ye, Chuanxiong Guo, and Danyang Zhuo. Collie: Finding performance anomalies in {RDMA} subsystems. In *USENIX NSDI*, 2022.

[25] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The QUIC transport protocol: Design and internet-scale deployment. In *ACM SIGCOMM*, 2017.

[26] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. FlowRadar: A better NetFlow for data centers. In *USENIX NSDI*, 2016.

[27] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. {RunD}: A lightweight secure container runtime for high-density deployment and high-concurrency startup in serverless computing. In *USENIX ATC*, 2022.

[28] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *IEEE ICDM*, 2008.

[29] Xusheng Luo, Luxin Liu, Yonghua Yang, Le Bo, Yuanpeng Cao, Jinghang Wu, Qiang Li, Keping Yang, and Kenny Q Zhu. Alicoco: Alibaba e-commerce cognitive concept net. In *ACM SIGMOD*, 2020.

[30] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. Pivot tracing: Dynamic causal monitoring for distributed systems. In *SOSP*, 2015.

[31] Sebastiano Miano, Fulvio Risso, Mauricio Vásquez Bernal, Matteo Bertrone, and Yunsong Lu. A framework for ebpf-based network functions in an era of microservices. *IEEE*

Transactions on Network and Service Management, 18(1):133–151, 2021.

[32] 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 *ACM SIGCOMM*, 2021.

[33] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, et al. Ananta: Cloud scale load balancing. *ACM SIGCOMM Computer Communication Review*, 43(4):207–218, 2013.

[34] Larry Peterson, Tom Anderson, Sachin Katti, Nick McKeown, Guru Parulkar, Jennifer Rexford, Mahadev Satyanarayanan, Oguz Sunay, and Amin Vahdat. Democratizing the network edge. *ACM SIGCOMM Computer Communication Review*, 49(2):31–36, 2019.

[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 *USENIX NSDI*, 2015.

[36] Mohammad Rajiullah, Per Hurtig, Anna Brunstrom, Andreas Petlund, and Michael Welzl. An evaluation of tail loss recovery mechanisms for tcp. *ACM SIGCOMM Computer Communication Review*, 45(1):5–11, 2015.

[37] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008.

[38] 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 *ACM IMC*, 2018.

[39] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C Snoeren. Passive realtime datacenter fault detection and localization. In *USENIX NSDI*, 2017.

[40] Zhiming Shen, Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert Van Renesse, and Hakim Weatherspoon. X-containers: Breaking down barriers to improve performance and isolation of cloud-native containers. In *ASPLOS*, 2019.

[41] Robin Sommer and Anja Feldmann. Netflow: Information loss or win? In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.

[42] Zhuo Song, Jiejian Wu, Teng Ma, Zhe Wang, Linghe Kong, Zhenzao Wen, Jingxuan Li, Yang Lu, Yong Yang, Tao Ma, Zheng Liu, and Guihai Chen. Zero+: Monitoring large-scale cloud-native infrastructure using one-sided rdma. *IEEE/ACM Transactions on Networking*, pages 1–16, 2024.

[43] Praveen Tammana, Rachit Agarwal, and Myungjin Lee. Simplifying datacenter network debugging with PathDump. In *USENIX OSDI*, 2016.

[44] 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 *USENIX NSDI*, 2019.

[45] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *EuroSys*, 2020.

[46] Marcos AM Vieira, Matheus S Castanho, Racyus DG Pacífico, Elerson RS Santos, Eduardo PM Câmara Júnior, and Luiz FM Vieira. Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. *ACM Computing Surveys (CSUR)*, 53(1):1–36, 2020.

[47] Weitao Wang, Xinyu Crystal Wu, Praveen Tammana, Ang Chen, and TS Eugene Ng. Closed-loop network performance monitoring and diagnosis with SpiderMon. In *USENIX NSDI*, 2022.

[48] Zhe Wang, Teng Ma, Linghe Kong, Zhenzao Wen, Jingxuan Li, Zhuo Song, Yang Lu, Guihai Chen, and Wei Cao. Zero overhead monitoring for cloud-native infrastructure using RDMA. In *USENIX ATC*, 2022.

[49] Minlan Yu, Albert Greenberg, Dave Maltz, Jennifer Rexford, Lihua Yuan, Srikanth Kandula, and Changhoon Kim. Profiling network performance for multi-tier data center applications. In *USENIX NSDI*, 2011.

[50] Xiao Yu, Pallavi Joshi, Jianwu Xu, Guoliang Jin, Hui Zhang, and Guofei Jiang. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. In *ASPLOS*, 2016.

[51] Chaoliang Zeng, Layong Luo, Teng Zhang, Zilong Wang, Luyang Li, Wenchen Han, Nan Chen, Lebing Wan, Lichao Liu, Zhipeng Ding, et al. Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing. In *USENIX NSDI*, 2022.

[52] Xiantao Zhang, Xiao Zheng, Zhi Wang, Hang Yang, Yibin Shen, and Xin Long. High-density multi-tenant bare-metal cloud. In *ASPLOS*, 2020.

[53] Zhuo Zhang, Chao Li, Yangyu Tao, Renyu Yang, Hong Tang, and Jie Xu. Fuxi: a fault-tolerant resource management and job scheduling system at internet scale. In *Proceedings of the VLDB Endowment*, volume 7, pages 1393–1404, 2014.

[54] Hao Zhou, Ming Chen, Qian Lin, Yong Wang, Xiaobin She, Sifan Liu, Rui Gu, Beng Chin Ooi, and Junfeng Yang. Overload control for scaling wechat microservices. In *ACM SoCC*, 2018.

[55] Jianer Zhou, Qinghua Wu, Zhenyu Li, Steve Uhlig, Peter Steenkiste, Jian Chen, and Gaogang Xie. Demystifying and mitigating tcp stalls at the server side. In *ACM CoNEXT*, 2015.

[56] Yu Zhou, Chen Sun, Hongqiang Harry Liu, Rui Miao, Shi Bai, Bo Li, Zhilong Zheng, Lingjun Zhu, Zhen Shen, Yongqing Xi, et al. Flow event telemetry on programmable data plane. In *ACM SIGCOMM*, 2020.

[57] 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 *CoNEXT*, 2022.

[58] Danyang Zhuo, Kaiyuan Zhang, Yibo Zhu, Hongqiang Harry Liu, Matthew Rockett, Arvind Krishnamurthy, and Thomas Anderson. Slim: OS kernel support for a low-overhead container overlay network. In *USENIX NSDI*, 2019.