SkyNet: Analyzing Alert Flooding from Severe Network Failures in Large Cloud Infrastructures

Bo Yang*, Huanwu Hu*, Yifan Li*, Yunguang Li, Xiangyu Tang, Bingchuan Tian, Gongwei Wu, Jianfeng Xu, Xumiao Zhang, Feng Chen, Cheng Wang, Ennan Zhai, Yuhong Liao, Dennis Cai, Tao Lin

Alibaba Cloud

Abstract

For providers operating large-scale global networks, the timeliness of network failure recovery significantly affects the reliability of network services. Ideally, a network monitoring system should have enough coverage to detect even minor issues, but high coverage means alert floods during severe network failures. In practice, there is a gap between the flooding raw alerts data collected by network monitoring tools and the readable information needed for failure diagnosis. Existing solutions using limited network monitoring data sources and heuristic diagnostic rules, lack comprehensive coverage and the capability to address severe failures, especially which network operators have never handled a similar one before. This paper presents SkyNet, a network analysis system to extract scope and severity information from alert floods. SkyNet ensures comprehensive coverage by integrating multiple monitoring data sources through a uniform input format, enhancing extensibility for new network monitoring tools. During alert floods, SkyNet groups alerts, assesses their severity, and filters out insignificant ones to aid network operators in mitigating network failures. To date, SkyNet has been running stably on our network for one and a half years without any false negatives and has successfully reduced the time-to-mitigation for over 80% of network failures since its deployment in production.

CCS Concepts

• **Networks** → *Data center networks*; **Network management**; **Network monitoring**; *Network reliability*.

Keywords

Data center networking, Network management, monitoring and troubleshooting

ACM Reference Format:

Bo Yang*, Huanwu Hu*, Yifan Li*, Yunguang Li, Xiangyu Tang, Bingchuan Tian,, Gongwei Wu, Jianfeng Xu, Xumiao Zhang, Feng Chen, Cheng Wang,, Ennan Zhai, Yuhong Liao, Dennis Cai, Tao Lin, *Alibaba Cloud*. 2025. SkyNet: Analyzing Alert Flooding from Severe Network Failures in Large Cloud Infrastructures. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3718958.3750536

*Bo Yang, Huanwu Hu and Yifan Li are the co-primary authors.

SIGCOMM '25, Coimbra, Portugal

ACM ISBN 979-8-4007-1524-2/25/09
https://doi.org/10.1145/3718958.3750536

1 Introduction

As a leading cloud provider, Alibaba Cloud provides 24×7 cloud network services to millions of customers worldwide. This service is supported by a network infrastructure comprising a wide area network (WAN) and numerous interconnected data centers. This infrastructure includes millions of servers, and the number of network devices reaches $O(10^5)$. Our network operators diligently maintain network reliability through careful design, construction, and verification processes. However, incidents in such a complex network are inevitable. As illustrated in Figure 1, the causes of network failures are diverse, making it challenging for our network operators to diagnose and address them. This highlights the critical need for effective monitoring and failure localization methods to minimize the impact of these incidents.

Existing research has significantly enhanced network monitoring capabilities. Nevertheless, most studies depend on a single or limited set of data sources, rendering them effective only under specific circumstances or for minor failures. For example, tools based on `Ping` can accurately identify a fault when a solitary link lacking a backup malfunctions. Traditional monitoring tools such as `SNMP` can detect packet loss within seconds, and `Syslog` collectors can provide instant notifications for failures logged by network devices. However, this reliance on a single data source frequently leads to restricted coverage. For example, `Syslog` cannot address routing errors that do not trigger runtime errors on a device, and other tools also exhibit their coverage limitations. Based on our industrial experience, the failure detection coverage of various network monitoring tools ranges from 3% to 84% (Figure 3). None can detect all network failures.

Here arises a natural question: *why not integrate all these tools to address the coverage issue and achieve comprehensive network monitoring?* Ideally, if the network operates smoothly or suffers only from minor failures, as depicted in Figure 2a, the monitoring system would generate few alerts, thus enabling operators to more easily identify the root cause. For example, some systems employ belief networks to automate failure recovery using Standard Operating Procedures (SOPs) built on prior experience [40, 49]. We refer to failures that network operators have previously encountered and matched with predefined rules as "known failures".

However, when addressing *severe* failures, as shown in Figure 2b, which occur only a few times globally each year [1] yet account for the majority of financial losses and reputation damage, merely collecting alerts from all tools presents the following challenges:

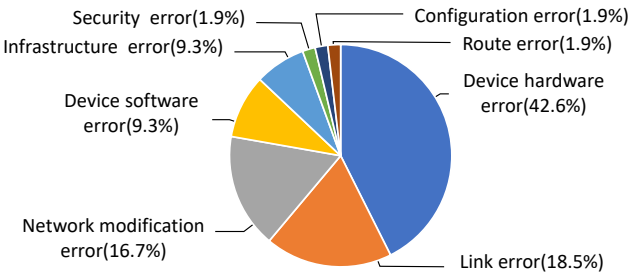

Figure 1: The proportion of network failure root causes.

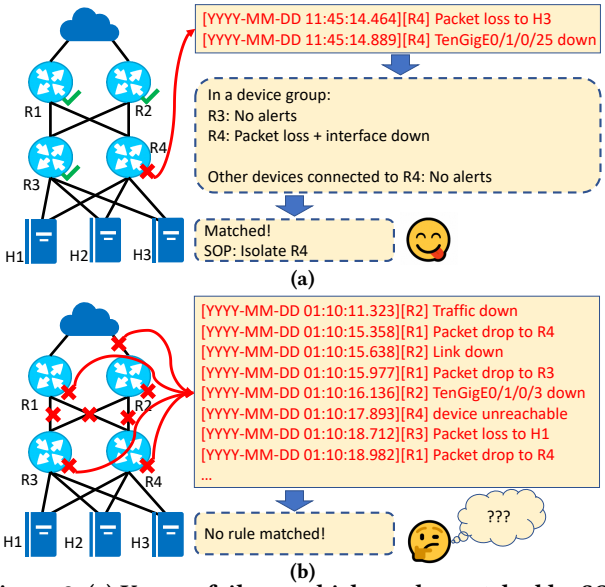

Figure 2: (a) Known failures which can be matched by SOP rules. (b) Unknown failures.

- Firstly, severe failures typically impact numerous devices and connections, generating an overwhelming number of alerts, making it difficult for operators to meet the minute-level recovery Service Level Agreement (SLA) requirements.
- Secondly, severe failures are rare, and an increasing number of them are unprecedented. We term these "unknown failures". Thus, reliance solely on predefined SOPs or models becomes impractical, leading to extended processing times or higher chances of error.
- Finally, there may be alerts triggered by minor failures or scheduled updates occurring concurrently, further complicating manual localization efforts.

Figure 4 illustrates the complete life cycle of detecting and resolving a network failure. A significant gap exists between the raw alerts generated by network monitoring tools and the actionable insights required for diagnosing the failure. In instances of severe failures, our network operators are inundated with alerts from the monitoring system, making comprehensive manual analysis challenging. As a result, operators often either overlook the root cause

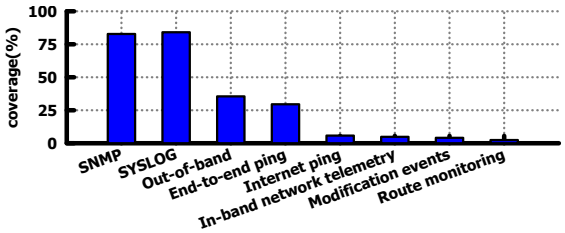

Figure 3: Network failure coverage of monitoring tools.

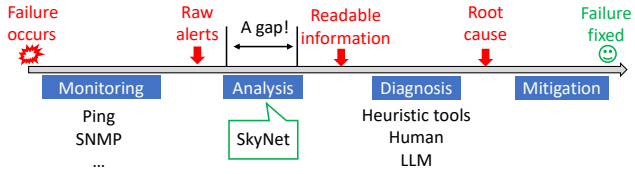

Figure 4: The life cycle of a network failure.

or take ineffective actions based on incomplete information, leading to unnecessary delays in localizing and mitigating the failure. Consequently, the recovery process is prolonged, often taking over an hour. Although this recovery time might appear rapid considering the scale of the failure, it remains unsatisfactory, as we are committed to providing minute-level failure recovery.

In this paper, we address the challenges of resolving severe failures in large-scale cloud networks. Our goals include enhancing alert correlation, prioritizing incidents (defined as a set of alerts originating from the same root cause), and root cause zooming-in, facilitating effective and efficient recovery from network failures.

To this end, we built SkyNet, an integrated network failure analysis system bridging the gap between monitoring and diagnosis. SkyNet employs a structured methodology: (1) It pre-processes diverse data sources using sophisticated algorithms. The system records all alerts' timestamp and location data, classifying them into predefined categories through heuristic or automatic methods. SkyNet then filters out non-essential alerts and aggregates specified alerts into comprehensive alerts based on experiential rules. (2) SkyNet constructs a hierarchical *alert tree*(refer to Figure 5c), indexed by location and comprising all collected alerts, maximizing the use of available data sources. This structure is then analyzed to identify locations impacted by incidents. (3) SkyNet employs telemetry tools to identify the failure location precisely. It queries user and traffic data related to the failure site and prioritizes concurrent alerts by urgency, enabling operators to address the most critical issues first.

SkyNet has already been deployed in our cloud infrastructure. Over the past one and a half years, SkyNet has enhanced network reliability for our operators. We present four representative cases to demonstrate its effectiveness and undertake extensive performance evaluations to assess SkyNet from multiple perspectives. The average mitigation time for severe failures decreased by 80%.

2 Background and Motivation

Alibaba Cloud operates a global network infrastructure. By July 2025, our network includes a private wide-area network and 89

Table 1: Existing network monitoring tools and their data sources.

Tools	Used in Production	Data Sources
RD-Probe [10]	True	Ping
Pingmesh [15]	True	Ping
NetNORAD [3]	True	Ping
deTector[34]	False	Ping
Dynamic mining [50]	True	Syslog
007 [7]	True	traceroute
Roy *et al.* [36]	True	INT
Netbouncer [42]	True	INT
PTPMesh [35]	False	PTP
Shin *et al.* [37]	False	SNMP
Redfish-Nagios [6]	True	Out-of-band

data centers across 29 geographic regions, covering North America, Europe, Asia, and Oceania [2]. The network comprises approximately $O(10^5)$ network devices. Failures in these devices and links can compromise service reliability and potentially breach SLAs.

Although it is impossible to entirely prevent network failures in an infrastructure of this magnitude, prompt recovery is crucial. Initially, we considered failure recovery to consist of only three steps: monitoring, diagnosis, and mitigation.

Over time, we have developed many advanced network monitoring tools for our in-production network. These tools are highly efficient, facilitating the rapid and precise identification of network failures. Additionally, simple network issues can often be resolved automatically. For instance, if a device exhibits packet loss while another device in the same group functions normally, and the traffic remains manageable, the malfunctioning device is automatically isolated.

However, this scenario is an ideal theoretical situation. In practice, not all network failures can be resolved so easily.

2.1 Limitation of Existing Work

This section examines the limitations of various monitoring data sources, which stem from a current focus on enhancing the functionality of tools reliant on individual sources. While these efforts improve performance and accuracy, a singular data source approach limits the ability to capture all network failures. The detailed limitations of each monitoring data source are outlined below:

- `Ping` is restricted to detecting failures related to reachability and cannot identify issues such as partial link failures that affect redundancy but not reachability.
- `Traceroute` loses effectiveness in networks with asymmetric paths or when tunnels such as Segment Routing Traffic Engineering (SRTE) are employed.
- `Syslog`-based tools cannot detect failures that are not evident to the devices, including silent packet loss, transmission errors, routing errors, or bit flips.
- `SNMP` [39] collects only information available within the SNMP protocol constraints.
- `Out-of-band monitor` addresses predominantly infrastructure related issues, focusing on device liveness, CPU utilization, temperature, etc.
- `In-band network telemetry` (INT) is not universally supported across all devices.
- `Route monitoring` is limited to the control plane and cannot diagnose data plane issues.

Table 1 provides a summary of several existing studies, indicating whether they are claimed to be used in production and specifying their data sources to highlight their limitations. Enhancing the comprehensiveness of our monitoring system necessitates integrating alerts from multiple network monitoring data sources.

Does combining multiple data sources work? Several attempts have been made to integrate multiple network monitoring data sources [12, 28, 29]. They employ a wide array of monitoring tools to trigger telemetry tasks or redirect network alerts to appropriate network operator teams. But essentially, these tools are doing a trade-off between achieving higher monitoring coverage and tolerating a larger number of alerts, which means a higher level of analytical difficulty. They cannot help avoid the problem of "alert flooding", especially when addressing "severe failures".

2.2 Alert Flooding in Severe Failures

"Severe failure" refer to incidents that significantly impact extensive areas, generating numerous alerts from various network monitoring tools almost simultaneously. This situation impedes our network operators' ability to prioritize critical alerts and identify the root cause of the failure.

As shown in Figure 2b, on one occasion, half of the cables serving as the Internet entry point for one of our data centers failed simultaneously, triggering a surge of alerts. Our `Syslog`-based tool issued multiple alerts concerning links and interfaces being down. Concurrently, `Ping`-based monitoring tools reported packet drops to specific servers in the data center, later attributed to congestion. Simultaneously, `SNMP` reported sharp traffic declines on certain links. Furthermore, the `Out-of-band monitor` flagged some network devices as unreachable. In total, over 10,000 alerts related to this network failure were generated within minutes. Additionally, unrelated glitches continued to produce alerts, further complicating the task of accurately identifying the root cause.

At that time, our network operators observed packet loss on several devices, initially attributing the issue to device-level failures. In accordance with standard operating procedures, the affected devices were isolated; however, this action failed to resolve the issue. Subsequently, the operators analyzed additional alerts and suspected the problem might be due to damaged cables. If this were the case, resolution would require a repair technician's intervention. Fortunately, a thorough inspection of each device and a detailed analysis of network packet propagation revealed that the packet loss was due to network congestion rather than faulty cables, suggesting that some cables remained functional.

To mitigate the congestion, bandwidth for certain services was reduced, allowing for the migration of some network services outside the data center. This mitigation process took several hours and resulted in significant financial and reputational losses. Notably, a congestion alert had been issued by our `SNMP` agent at the onset of this incident, but it was obscured by a flood of alerts. Reflecting on this challenging experience, we have decided to implement a network analysis system to extract valuable insights from a multitude of alerts collected by monitoring tools, thereby expediting the recovery from severe failures.

2.3 Why not LLM?

Recently proposed network diagnostic systems leveraging large language models (LLMs), such as Ahmed *et al.* [5], MonitorAssistant [54] and NetAssistant [44], theoretically fails to meet our requirements. The primary challenge is the vast volume of alert data generated by monitoring tools. For instance, even only considering `Syslog`, approximately 10 million entries are produced every 15 minutes, far surpassing the input capacity of existing long-context LLMs, which at most support 20 million tokens [11]. Thus, these LLM-based solutions necessitate the specification of time and location parameters, operating more as diagnostic tools rather than comprehensive monitoring systems. Incorrectly chosen parameters can render the LLM incapable of accurately identifying the root cause of issues.

Additionally, LLMs function as black boxes. Although they might efficiently identify root causes under ideal conditions, their propensity for hallucination [20] means that erroneous diagnoses could force network operators to restart the diagnostic process, potentially wasting time on misleading instructions. As a network service provider, it is essential to maintain control even in worst-case scenarios, a reliability that LLMs currently lack. In summary, LLMs are great future opportunities, but for now, they serve only as supplementary tools for evaluating network failures, necessitating manual intervention for severe issues.

2.4 The Desire of Operators

To assist network operators in addressing severe network failures manually, it is crucial to understand the specific information they require. Our network operators possess significant experience and can swiftly diagnose the root cause of network failures when the number of alerts is limited to no more than approximately 20. In a practical scenario, our network monitoring tools generated 16 alerts within five minutes. Among these, an internet unreachability alert prompted the operators to address the issue. Upon initial inspection, they identified that 12 alerts originated from the same location, while the remaining 4 were unrelated. Their attention was subsequently drawn to associated `Syslog` alerts regarding runtime errors, ultimately determining that a software error caused the failure, which was subsequently reported to the device vendor.

Based on the above cases, when a network failure occurs, it is imperative for on-call network operators to immediately recognize its presence. The manual process of identifying failures involves selecting all relevant alerts to determine if any indicate potential network issues, such as packet loss, high transmission delay, or bit flips. During this process, network operators are primarily concerned with the presence of certain alert types rather than their quantity. This procedure is laborious, and, in practice, network failures are often detected by customers and reported through complaints.

Once a network failure is detected, network operators need to diagnose the issue. Unlike the detection phase, diagnosing requires identifying the presence of root cause alerts, including link failures, device malfunctions, and hardware or software error logs. The redundancy design in our data center may prevent these root-cause alerts from impacting customers. However, if customers are affected, network operators must diagnose the failure based on these alerts. Furthermore, if multiple network failures occur simultaneously, operators must prioritize addressing the most severe issue first.

In conclusion, to address network failures manually, network operators require the following information: **the occurrence of network failures, related alerts, specifically the types of alerts, and the severity of these failures.** Presenting a distilled set of approximately 10 messages containing this information instead of the original $O(10^4)$ alerts enables operators to manually control network failures.

2.5 Our Goal

Our goal is to develop an in-production network analysis system to aid network operators in detecting, displaying, and evaluating all network failures.

Detection. The system should cover all types of network failures. Upon the occurrence of any network failure, it should be detected and reported to the network operator immediately, even in cases of previously unknown failures.

Display. The system must pinpoint the time and location of network failures, efficiently extracting and presenting readable information from numerous alerts triggered by severe network incidents. Additionally, it should classify all related network alerts.

Evaluation. In situations where multiple network failures occur simultaneously, the system should evaluate the urgency of each failure to help operators prioritize their response. Ideally, the system should also identify the specific device or link requiring repair.

3 Overview

We introduce SkyNet, a comprehensive network analysis system designed for global network infrastructures. The central concept of SkyNet is aggregating network alerts from numerous monitoring tools and organizing them into clusters by time and location. We refer to these clusters as *incidents*, which categorize the alerts according to type. Subsequently, incidents are prioritized based on severity, determined by associated traffic and customer data.

The architecture of the SkyNet system is presented in Figure 5a. SkyNet incorporates various data sources as input, and previous research ensures the clarity and precision of these sources. Alerts from these data sources include timestamps and location details. Using a combination of heuristic and automated techniques, each alert is assigned a specific type, thus characterizing it by timestamp, location, and type.

Utilizing the structured data, we initially filter alerts that align with historical patterns and automatically apply SOPs. Subsequently, SkyNet implements incident discovery to aggregate alerts into incidents, categorizing them by type. SkyNet then prioritizes the incidents based on their traffic impact on the affected network and recommends the order in which they should be addressed. Figure 6 illustrates detailed examples of incidents identified by SkyNet. During the development and adjustment phases, it became evident that utilizing sufficient data sources along with a simple algorithm enables both high levels of accuracy and performance.

Historically, when network failures occurred, on-call network operators were required to manually sift through a deluge of alerts. They needed to filter out extraneous alerts and identify all relevant ones. Given the substantial volume of alerts and limited human processing capacity, this process was not only time-consuming but also prone to errors. SkyNet automates this filtering and clustering

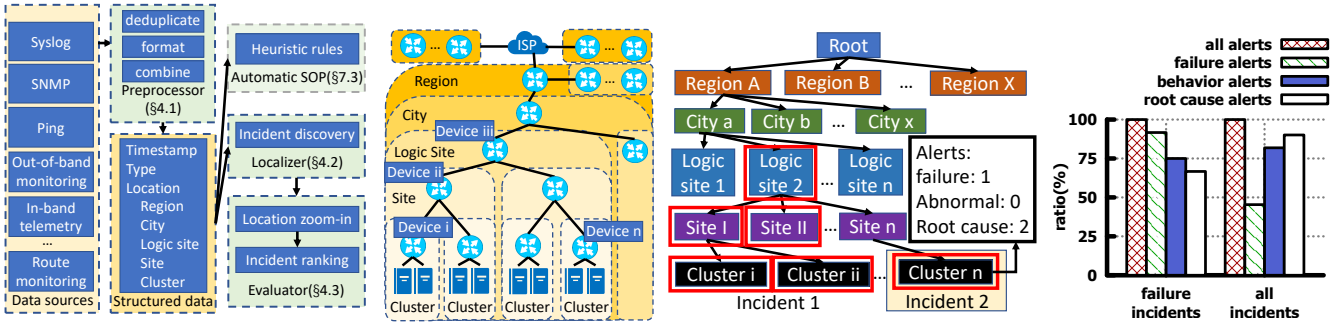

Figure 5: (a) System overview of SkyNet. (b) The hierarchy of our cloud network. (c) Hierarchical alert tree. (d) Correlation between incidents and different alerts.

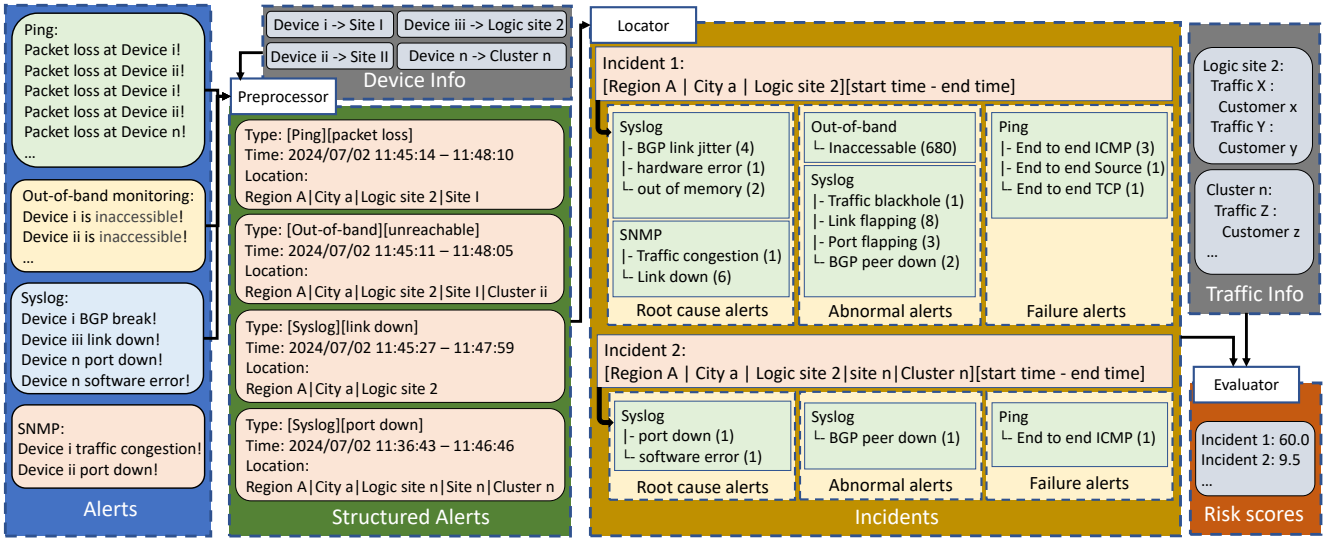

Figure 6: A running example showing how SkyNet processes alerts, groups alerts into incidents and ranks incidents by computing their risk scores.

process, assisting network operators in addressing the most critical network failures first. This automation allows operators to concentrate on root cause analysis of significant network issues without being overwhelmed by irrelevant alerts. This transformative advancement in network failure recovery has markedly decreased our average recovery time.

4 Design

In this section, with a running example(shown in Figure 6), we introduce how SkyNet works by three functional modules including the preprocessor (§4.1), the locator (§4.2) and the evaluator (§4.3).

4.1 Preprocessor

The original data sources utilized by SkyNet are detailed in Table 2. While the output alerts collected from network monitoring tools are typically well-structured, an alert ideally indicates when, where, and what is occurring in the network. However, even with the assumption that all monitoring tools are well designed, the raw data collected from them cannot be directly employed by SkyNet. The differences between monitoring tools are highlighted as follows:

- Alert generation frequency varies among monitoring tools. For instance, in our practice, `Ping` outputs one data point every 2 seconds, whereas `Syslog` produces a log only upon encountering an error. To equitably prioritize different monitoring tools, alert frequencies must be normalized.
- The structure of location information differs. For example, the source device for `Syslog` and `SNMP` alerts is typically evident. Conversely, other data sources, such as packet loss between two logical sites, are generally attributable to an intermediary link. In such cases, the ping tool reports packet loss alerts for the affected link.

Table 2: Network monitoring tools used by SkyNet.

Data source	Description
Ping statistics	Periodically records latency and reachability between pairs of servers
Traceroute statistics	Periodically records latency of each hop between pairs of servers
Out-of-band monitoring	Periodically collect device information from out-of-band, including liveness, CPU and RAM usage, etc.
Traffic statistics	Data from traffic monitoring system `sFlow` and `netFlow`.
Internet telemetry	Data from a monitoring system that ping Internet addresses from DC servers.
Syslogs	Errors detected by network devices.
SNMP&GRPC	Standard network protocol, including status and counters of interfaces, RX errors, CPU and RAM usage.
In-band network telemetry	Sending test packets and collect information of devices bypassed.
PTP	System time of network devices out of Synchronization.
Route monitoring	Loss of default/aggregate route, route hijack and route leaking.
Modification events	Failure of network modification triggered automatically or manually.
Patrol inspection	Run manual defined commands on network devices and collect result periodically.

- Alert contents are varied. `Syslog`, for example, has thousands of types of command-line interface outputs. Furthermore, distinct monitoring systems present different alert contents.

Given the potential integration of new monitoring tools in the future, it is crucial to ensure the system's extensibility by defining a uniform input format and converting raw alert data into this format. SkyNet addresses the disparities among various data sources by transforming raw data into standardized data structures through the following methods.

To address the variation in frequency among different monitoring tools, the preprocessor aggregates similar alerts based on prior experience. For instance, SkyNet uses the start time of packet loss detected by ping as the alert timestamp, with subsequent alerts contributing to a "duration" attribute that aids network operators.

To explain how the location issue is resolved, prior knowledge must be introduced. The entire network, comprising both the WAN and the DC network, is organized hierarchically, as shown in Figure 5b. Each device is assigned a level in this hierarchy. Based on our observations, if a failure occurs on a device, all devices at the lower levels are impacted, which helps evaluate the extent of a network failure. Therefore, we associate alert information with the hierarchical level of the device from which the alert is collected. An alert related to a link is split into two alerts corresponding to the devices it connects. In the example illustrated in Figure 6, the alerts from Devices i, ii, iii, and n are associated with Site I, Site II, Logic Site 2, and Cluster n, respectively.

For alerts collected from various network monitoring tools, it is essential to separately consider the alert types. For tools with limited alert content, such as `Ping` for monitoring packet loss rates and `SNMP` for traffic monitoring, alert types are manually defined. To process `Syslog`, templates are employed to automatically convert command-line outputs into alert types. The template tree is constructed as follows: initially, it gathers command-line outputs from all devices and breaks them down into individual words. Variable words, such as addresses, interfaces, and numbers, are then removed using predefined regular expressions. The remaining words create templates for alert classification. An FT-tree [56] is constructed using these templates, based on their frequency. The classification process starts with manually assigning types to existing alerts. With hundreds of alert types to consider, we prioritize the most critical and complete the manual classification over several months. In practice, although severe failures are rare and unprecedented, these templates account for `Syslog` alerts during such events.

After standardizing the data structure of the alerts, we encounter another challenge: the sheer volume of alerts. While each individual monitoring tool produces reasonable alerts, from the perspective of SkyNet, which evaluates the entire network to identify hotspots, some alerts necessitate further processing. SkyNet primarily employs three methodologies to reduce the number of alerts.

- Consolidate Identical Alerts: SkyNet may receive duplicate alerts from the same monitoring tool. For instance, if the CPU or RAM usage of network devices is excessively high, or if there is substantial traffic through an interface, `SNMP` might repeatedly generate identical alerts. If a specific alert has been analyzed, SkyNet disregards the subsequent ones. Otherwise, it updates the timestamp of the initial alert.
- Consolidate Alerts from a Single Data Source: Certain alerts gain importance only when they exceed a specific threshold. For example, sporadic packet loss is ignored, while persistent packet loss is recorded. Moreover, some alerts are linked to associated alerts; for example, a sudden traffic surge on an interface may cause increased traffic through adjacent interfaces. SkyNet filters out these related alerts to streamline the analysis.
- Consolidate Alerts from Diverse Data Sources: Alerts from various monitoring tools may not independently indicate a problem; however, they might suggest an issue when combined with data from another tool. For example, a sudden decrease in port traffic is typically expected, but if it occurs alongside packet loss or a `Syslog` error, it might suggest an abnormal decline in traffic.

Ultimately, the preprocessor generates a series of filtered, structured alerts that include information such as alert type, time, and location.

4.2 Locator

When a network failure occurs, identifying the root cause begins with detecting the failure and understanding which alerts are associated with it. Prior to the development of SkyNet, network failure mitigation typically involved responding to custom complaints, typical alert patterns, or alert floods. Subsequently, network operators inspect the links and devices that generated alerts or that are topologically adjacent. This involves connecting to the devices, reviewing logs, and executing status commands in a systematic manner to identify and rectify the root cause. Based on our network operating experience, we have gathered the following insights:

- Multiple alerts appearing at the same location within a short timeframe may indicate a network incident. A fundamental approach is to group alerts by both their time and location.
- Network alerts often propagate through topological links; if two adjacent devices generate alerts simultaneously, they likely share the same root cause.
- It is crucial for network operators to comprehend the full extent of a network failure in order to evaluate its scope and severity.

Utilizing the aforementioned insights, the proposed system, referred to as SkyNet, employs a hierarchical alert tree structured by the temporal and locational characteristics of alerts, known as the "main tree" (illustrated in Figure 5c). When a new alert is received, SkyNet checks for the existence of its location node; if found, the alert is added to that node. Otherwise, a new node is created for the alert(Algorithm 1). SkyNet continuously monitors these nodes, removing alerts that have expired, and assesses whether any nodes contain a sufficient number of alerts. The timeout threshold should be minimized to avoid associating unrelated alerts. However, if this threshold is set shorter than the alert delay, other alerts under the same node might expire before the delayed alert arrives. Considering that the maximum delay for alerts, due to SNMP on older network devices with CPU limitations, is approximately 2 minutes, we have selected a 5-minute threshold to prevent data gaps caused by transmission errors or high CPU usage, which could lead to a maximum delay of approximately 4 minutes.

Algorithm 1: Alert node insertion

```
Data: alert data d, main tree m, incident trees I
Result: updated main tree M and incident trees I
for i ∈ I do
    if d.location ∈ i then
        i.get(d.location).add(d);
        i.updateTime = d.timestamp
    else
        if d.location ∈ i.subtree then
            i.put(d.location, list());
            i.get(d.location).add(d);
            i.updateTime = d.timestamp
if d.location ∈ m then
    i.get(d.location).add(d);
else
    i.put(d.location, list());
    i.get(d.location).add(d);
```

The alert count for a node encompasses its own alerts as well as those of its descendant nodes. If the number of alerts surpasses a specified threshold, the subtree beneath the node is replicated from the main tree and designated as an "incident tree". However, as network alerts propagate through topology links, when calculating alerts, the algorithm only considers alerts within the area connected to the root node of the incident tree(Algorithm 2). For instance, in Figure 5c, alerts are raised solely by the nodes enclosed in red rectangles. Since device n is isolated from other alerting nodes in the main tree, the alerts collected from device n are attributed to a different root cause. Consequently, two incident trees are stored. Subsequently, when a new alert arrives, the algorithm evaluates both the main tree and any existing incident trees to determine the appropriate location for the alert. If an incident tree for a node already exists, the algorithm refrains from creating a new one. If no alerts are added to an incident tree for an extended period, the incident tree times out. Theoretically, the timeout threshold is 5 minutes(Algorithm 3), but as timeliness is not critical here, the threshold is set to 15 minutes.

Algorithm 2: Tree generation

```
Data: main tree m, incident trees I
Result: updated main tree M and incident trees I
for node n ∈ m do
    if n ∈ I.roots then
        continue;
    failureAlerts, allAlerts = sum(n.subtree);
    if (failureAlerts > 2)||(allAlerts > 5)||
    (failureAlerts > 1&&allAlerts > 3)|| then
        for i ∈ I do
            if i.root ∈ n.subtree then
                I.remove(i);
        I.roots.add(n);
        n.subtree.updateTime(max(n.timestamps));
        I.add(n.subtree);
```

To determine whether there are sufficient alerts under a node, it is essential to consider two potential issues that may arise from merely counting the number of alerts associated with a node. First, the frequency of alerts from different data sources can vary. Some alerts, such as "link down", are triggered only once, while others, such as "packet loss", often appear in batches with diverse sources and destinations. Second, the significance of alerts may differ. For example, packet loss usually indicates that network service is compromised, whereas a sudden increase in delay might be only the result of concurrent access and could be automatically resolved.

To address the discrepancy in alert frequency, the proposed method counts alerts based on their types. If multiple alerts of the same type are added to a node, they are counted only once.

Algorithm 3: Tree checking

```
Data: main tree m, incident trees I
Result: updated main tree M and incident trees I
for node n ∈ m do
    if time.now > n.timestamp + 300 then
        m.remove(n);
for i ∈ I do
    if time.now > i.updateTime then
        i.end();
```

In addressing the issue of varying alert importance, SkyNet categorizes alerts into three levels:

- Failure alerts: These occur when network behavior is definitively abnormal, including instances of packet loss, packet bit flip, and high packet transmission latency.
- Abnormal alerts: These are triggered by irregular network packet behaviors such as jitter, sudden increases in latency, or abrupt decreases in flow. They do not directly imply network failures.
- Root cause alerts: These indicate failures of network entities, such as device or NIC failures, link outages, CRC errors, or risky routing paths.

To illustrate the differences and relationships among these alert types, consider the following example: Imagine a router in the network suddenly fails, redirecting its traffic through a backup router, leading to congestion and subsequent packet loss. Packet loss is absolutely faulty and thus classified as a failure alert. Concurrently, there is a sudden decrease in traffic on the failed router and a sharp increase on the backup router. These abrupt changes are classified into abnormal alerts as they might be expected due to user behavior. However, identifying packet loss and sudden traffic changes alone does not directly guide network operators in resolving the issues. It is the root cause alerts, such as the notification of a device failure, that specify the precise solution. Network operators then understand they must repair the malfunctioning device based on this alert.

As discussed in §2.4, failure alerts are considered the most authoritative during incident detection, while the others are regarded as less significant. Historical network failure records support the greater importance of failure alerts in incident detection. As illustrated in Figure 5d, although the overall incidence of failure alerts is relatively low, nearly all network failure incidents are accompanied by these alerts, highlighting their positive correlation with network failures.

Given this correlation, failure alerts are of paramount importance in incident detection. The threshold for incident tree generation is set at either two failure alerts, one failure alert plus two other alerts, or five alerts of any type. These thresholds were determined through experiments detailed in §6.3 and are consistent across all location layers. Although it might be assumed that higher-level thresholds should be set higher to filter noise from lower-level nodes, in practice, SkyNet often identifies a single alert, such as a port or device down, as the root cause for a complete failure. Consequently, each alert is treated with importance by maintaining uniform thresholds across layers.

Despite having equal significance in incident detection, abnormal alerts and root cause alerts require different approaches from our operators to resolve incidents. Specifically, root cause alerts are more critical during manual mitigation procedures. Therefore, SkyNet categorizes them into two levels to aid network operators.

In practice, random events can generate numerous alerts. For instance, if a device probe used for activity detection encounters an error, it may trigger numerous identical device down alerts. These alerts can easily surpass the incident threshold, causing false alarms and increasing the workload for our network operators. To mitigate this, we consolidate alarms of the same type from different devices into a single alert. This means we focus solely on the types of alarms occurring within an incident's scope, irrespective of their origin. This method reduces false positives, as demonstrated in §6.3.

The output of the locator, illustrated in Figure 6, displays the incidents. Each incident report includes the time and location of the incident, along with the alerts associated with it, categorized into three types.

4.3 Evaluator

In production settings, minimizing losses requires prioritizing the resolution of failures that impact the most crucial services. A straightforward approach is to analyze the affected workflows based on time frames and the locations of grouped incidents. However, simply grouping alerts into incidents provides network operators with information only about the occurrence of incidents, without indicating their severity. For instance, consider one link experiencing 5% packet loss and another link with 50% packet loss. Both can be monitored using tools like `Ping`, with packet loss alerts generated, yet they differ significantly in severity. When multiple incidents occur simultaneously, the network operator might not prioritize the most severe one accurately.

A real-world example from our network illustrates this issue. Two incidents happened concurrently at different locations. One incident affected a larger number of network devices, generating more alerts, whereas the other involved fewer devices but affected more critical customers. The network operator focused on resolving the incident with more alerts, causing a delay in addressing the second, more critical incident, resulting in unnecessary financial and reputational losses.

Thus, SkyNet provides a quantitative assessment of incident severity, prioritizing incidents to help network operators address the most critical issues first. The severity of incidents is evaluated based on the following criteria:

- Packet Loss Ratio: This includes packet loss ratios determined by both `Ping` and `sFlow`, as packet loss directly impacts service quality. We use the packet loss ratio to normalize the packet loss rate collected by `sFlow` under varying traffic conditions, rather than relying solely on the packet loss rate.
- Link Break Ratio: Devices connect to link sets for redundancy purposes. While a break in some links within a set may not directly cause packet loss, it can reduce the total bandwidth, thereby affecting service quality during traffic surges.
- Importance of Affected Customers: Customers who purchase services with higher stability expectations should experience minimal downtime. The impact on affected customers is determined using traffic data collected via `Netflow`.
- Duration of the Incident: Ignoring an incident indefinitely is unreasonable, even if its impact is mild. Adhering to our service level agreements (SLA), prolonged recovery times cause significant financial and reputational losses.

With the information above, the severity y_k of an incident k is calculated using Equations 1-3.

$$I_k = \max\left(1, \sum_{i=1}^{N} d_i g_i u_i + \sum_{j=1}^{N} l_j g_j u_j\right) \tag{1}$$

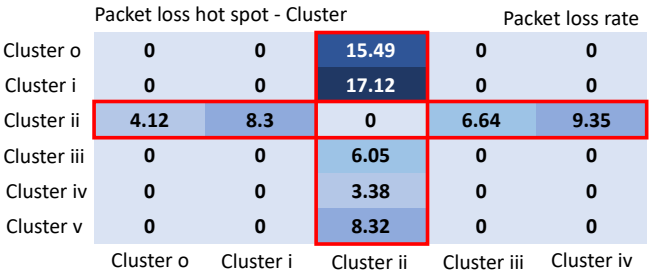

Figure 7: A reachability matrix example.

Table 3: Explanation of symbols used to evaluate incident.

Symbol	Explanation
N	The total num of circuit set related to the incident
d_i	The break ratio of circuit set i
l_i	The ratio of SLA flows beyond limit on circuit set i
g_i	The importance factor of customers related to circuit set i
u_i	The number of customers related to circuit set i
R_k	The average ping packet loss rate
L_K	The max average SLA flow rate beyond limit
ΔT_k	The alert lasting time
U_k	The number of important customers

$$T_k = \max\left(\log_{\frac{1}{R_k}}\left(\Delta T_k + Sig(U_k)\right), \log_{\frac{1}{L_k}}\left(\Delta T_k + Sig(U_k)\right)\right) \quad (2)$$

$$y_k = I_K \cdot T_k \quad (3)$$

Table 3 clarifies the symbols used in the equations, each corresponding to variables within the evaluated incident's context. For redundancy purposes, all links connecting network devices consist of multiple circuits, each is called a circuit set.

To make it easier to understand, we split the incident severity calculation into two parts. The first part is impact factor I_k, illustrating the impact on significant users, calculated by Equation 1. The impact factor I_k increases as the number of circuit sets used by important users that experience breaks or overloads increases. With a maximum function, we ensure that the severity remains non-zero, even when no critical customers are affected. The second part is time factor T_k, calculated by Equation 2. The time factor T_k demonstrates that severity escalates with the incident's duration, ensuring all incidents eventually capture the attention of network operators. An increased average packet loss rate accelerates this growth. The sigmoid function (represented by Sig) significantly influences severity when only a few key users are affected but stabilizes when many important users are impacted, thus preventing false alarms due to jitter. With this time factor T_k, critical incidents are promptly highlighted, ensuring attention while excluding incidents resolved by transient jitter.

Location zoom-in. To maintain manageable control over incident areas and prevent the average packet loss rate from being influenced by unrelated links, our method employs behavior monitoring tools to conduct a "location zoom-in". This approach aids in accurately identifying the exact location of the incident.

The location zoom-in is triggered under the following conditions:

- A focal point is identified within the reachability matrix calculated by the `Ping` tool. Using the end-to-end reachability telemetry results produced by `Ping`, a reachability matrix is constructed. For example, in the illustrative scenario, the reachability matrix, depicted in Figure 7, allows zooming into Cluster II. Source-destination pairs exhibiting high packet loss rates are indicated by dark colors. A column and a row shaded in dark colors pinpoint the precise location of the incident after zoom-in. The reachability matrix vary in granularity from cluster to region.
- The `sFlow` detects packet loss, with all affected devices tracing back to a node within the incident tree. Thus, the incident location is refined to this specific node.
- `In-band network telemetry` generate test flows with designated DSCP values to compare input and output rates. A discrepancy between these rates indicates packet loss. Should the affected devices correlate with a node in the incident tree, the zoom-in process identifies this as the incident location.

There may be instances where the location cannot be further refined. In such cases, emergency procedures revert to the general location of the incident. The evaluator's output assigns a severity score. In this running example, incident 1 has a severity score of 60, whereas incident 2 scores 9.5. Consequently, our network operators prioritize addressing incident 1.

5 Deployment Experience

SkyNet has operated stably within our global network infrastructure for one and a half years and detected $O(10)$ severe network failures.[1] It has expedited the processes of network failure detection and localization, ultimately reducing failure mitigation time. We present case studies demonstrating SkyNet's assistance to our network operators and examine its extensibility based on our deployment experiences.

5.1 Case Study

We demonstrate the capabilities of SkyNet by analyzing its performance in four incidents involving simple, concurrent, and severe failures.

Automatically SOP for known failures. As illustrated in Figure 2a, SkyNet detected various alerts from a network device, including packet loss, `Syslog` errors, and abnormal `SNMP` counter statistics. No other devices in the same group triggered alerts. Consequently, SkyNet generated an incident report exclusively for this device and initiated an automatic SOP, successfully isolating it. Our network operators later verified the correctness of the automatic isolation. The entire mitigation process was completed in approximately one minute without manual intervention.

Multiple scene detection. Last year, we experienced a DDoS attack targeting five different locations simultaneously. Without SkyNet, our network operators would have needed to manually correlate the abrupt traffic increases. By clustering the alerts by location, SkyNet generated five separate incidents, indicating the attacks were unrelated. With this information, our network operators quickly implemented IP blocking for all five regions efficiently and comprehensively. Without SkyNet, although manual diagnosis

[1]We omit the absolute numbers for confidential reasons.

can manage alert floods and identify a DDoS attack, it is possible to overlook certain attack points, thereby increasing recovery time and causing greater damage.

Scene ranking. On one occasion, two network failures occurred nearly simultaneously. SkyNet successfully detected both, generating two incidents. Although one incident covered a larger geographical area and included more alerts, SkyNet identified the other incident as more urgent based on the traffic and service importance. This prioritization enabled our network operators to address the more critical incidents first. Without SkyNet, network operators may be distracted by minor yet more conspicuous failures, potentially resulting in increased losses.

Fine-grained localization. Following the deployment of SkyNet, a similar failure occurred as described in §2.2, involving another broken Internet entrance cable. SkyNet consolidated the related alerts into a single incident and, using the reachability matrix derived from ping data, identified the data center entrance as the likely failure point. Relevant alerts, including link-down notices, were effectively grouped and displayed. The mitigation time was reduced to just a few minutes, including cable repairs, compared to several hours in the case outlined in §2.2, representing a near two-orders of magnitude reduction in recovery time.

5.2 Extensibility

The initial input format of SkyNet was limited to data sources such as `SNMP`, `Syslog`, and `device detection`. Over the past eight years, additional data sources, including `route monitoring`, `end-to-end ping`, `modification events`, and `GRPC`, have been iteratively incorporated. The seamless integration of these new data sources demonstrates the extensibility of SkyNet, suggesting its potential to accommodate even more data sources in the future.

6 Performance Evaluation

We conducted a series of experiments to evaluate SkyNet, aligning with the goals outlined in §2.5. Initially, we establish the necessity of integrating multiple data sources with SkyNet to achieve extensive network failure coverage (§6.1). Subsequently, we examine the preprocessor's impact on reducing alerts and evaluate its advantages by presenting the time required for incident detection (§6.2). Furthermore, we demonstrate the accuracy of the locator, including its false positive and false negative rates (§6.3). By adjusting the incident thresholds, we elucidate the criteria for their determination based on accumulated experience. We also compare the severity scores of all incidents with failure incidents, evidencing the evaluator's effectiveness in filtering trivial incidents (§6.4), thereby diminishing the workload of network operators. Finally, we deliver an overall assessment of SkyNet by comparing the network failure mitigation times pre- and post-deployment in our network.

Experiment setup. SkyNet was deployed on a server using an Intel Xeon Platinum 8260 CPU(2.40 GHz), with 96 cores and 32 GB of RAM for all performance evaluations. The dataset utilized for the evaluation comprised alerts collected from our real production network, as described in 1, consisting of $O(10^5)$ network devices, over the past one and a half years. The alerts are raised by over 10 network monitoring tools shown in Table 2

6.1 Coverage

Initially, we have demonstrated that no single network monitoring tool is sufficient to address all network failures, shown in Figure 3.

In our analysis, we systematically removed data sources, beginning with those having low coverage and progressing to those with high coverage. This allowed our network operators to manually evaluate SkyNet's false positives and false negatives. As depicted in Figure 8a, a reduction in the number of data sources minimally impacts false positives but leads to an increase in false negatives, pointing to potentially overlooked failures. While false positives mainly impose additional workload on network operators and can be tolerated within certain limits, it is vital to minimize false negatives. Thus, integrating as many data sources as possible is crucial to ensuring comprehensive coverage.

6.2 Preprocessing

The alert preprocessing occurs through a stream processing mechanism. Consequently, evaluating the time required to preprocess an individual alert is not meaningful. Instead, our evaluation emphasizes how effectively the preprocessing tool reduces the overall volume of alerts. We collect logs from all monitoring tools over several hours and process them using the preprocessing stream. On average, the system generates approximately 100,000 alerts per hour before preprocessing. The preprocessing tool typically reduces this number to fewer than 10,000 under normal conditions and to fewer than 50,000 in more extreme circumstances. The resulting performance benefits are substantial.

The locator leverages the preprocessor output to identify incidents hourly. In the worst-case scenario, it takes less than 10 seconds to locate failures, which is significantly below our minute-level Service Level Agreement (SLA). In addition, there is a positive correlation between the time required and the number of alerts. Without the preprocessor, the time to locate failures can extend to several minutes, which is ineffective for mitigating network issues.

6.3 Accuracy

We assessed the accuracy of SkyNet using various incident detection parameters. The results are depicted in Figure 9. The x-axis uses the format $A/B{+}C/D$, representing incident generation thresholds as "A failure alerts", "B failure alerts and C other alerts" or "D any alerts". In production, the parameters currently set for SkyNet are "2/1 + 2/5". A parameter set to 0 indicates the corresponding threshold is disabled. While some false positives bringing extra solving time can be tolerated, minimizing false negatives causing undetected failures is crucial and ideally should be eliminated.

Type and location. As mentioned in §4.2, SkyNet, by default, consolidates multiple alerts of the same type at different locations into a single alert. The "type and location" data refer to incident detection results treating alerts of the same type at different locations as distinct alerts. Although this approach avoids false negatives, it increases false positives from less than 20% to 70%, imposing an additional burden on network operators.

Disabling Thresholds. As noted in §4.2, all three thresholds are justified. Disabling any of these thresholds results in higher false negatives.

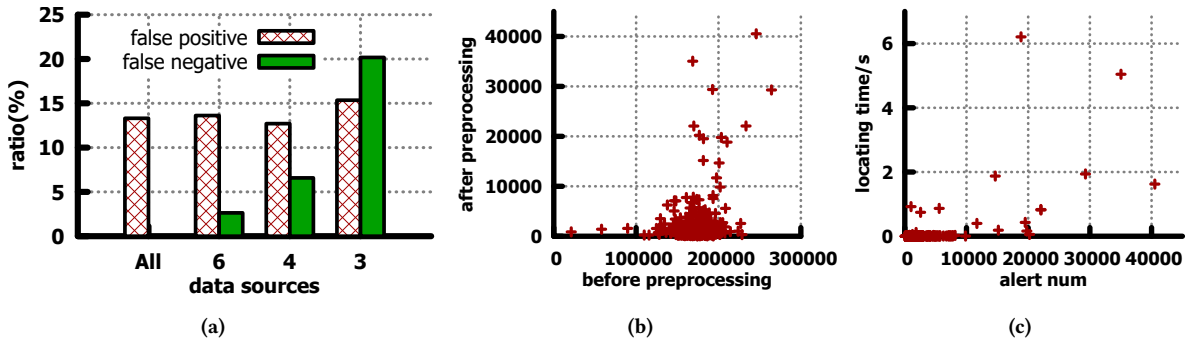

Figure 8: (a) Locating accuracy vs data source number. (b) Alert num before and after preprocessing. (c) The time cost of locating.

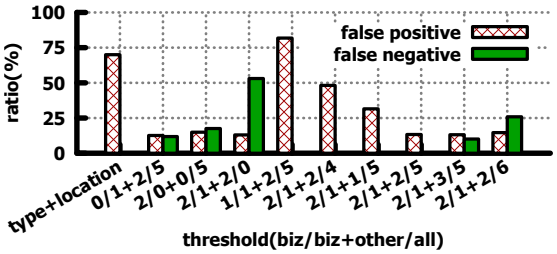

Figure 9: Accuracy with different parameters.

Adjusting parameters. We also experimented with adjusting the thresholds and determined that the current settings used by SkyNet provide the lowest false positives while maintaining zero false negatives, thereby achieving the optimal balance between false positives and false negatives.

6.4 Evaluator

Indeed, even when incidents are grouped by locator, hundreds of network events occur monthly, though only a few truly constitute harmful network failures. This volume can be overwhelming for network operators to manage. It should be noted that not all high-risk incidents culminate in network failures, thanks to our high-availability design, which incorporates redundant links and devices. Nevertheless, high-risk incidents are more likely to impact our customers. We have configured a severity threshold in the evaluator, allowing SkyNet to only trigger alerts for incidents whose severity exceeds this threshold. This threshold is determined based on historical experience. Over the past nine months, we gathered network incidents identified by SkyNet, then had our network operators select those attributable to network failures. The results are depicted in Figure 10a. For clarity, we capped the maximum severity score at 100. Generally, the severity scores for failure-related incidents are higher than those for all incidents. To avoid false negatives, we set the severity threshold score to 10, which reduces the number of incidents by almost two orders of magnitude, as illustrated in Figure 10b. After applying this filter—while ensuring no false negatives—the average number of incidents falls to fewer than one per day, which effectively alleviates the analytical workload of network operators regarding network failures.

During its nine-month stable operation, SkyNet has assisted our network operators in mitigating approximately ten failures. Compared to mitigation efforts without SkyNet, the time required for network failure resolution has significantly decreased. Figures on mitigation time statistics, both with and without SkyNet, are shown in Figure 10c. Both the median and maximum mitigation times have been reduced by more than 80% due to SkyNet.

The median mitigation time is reduced from 736s to 147s, and the maximum mitigation time is from 14028s to 1920s, but we cannot directly post these detailed numbers.

7 Lessons and Discussion

Here we share some experience on consolidating our network monitoring system, and what we are doing now to further improve SkyNet.

7.1 Visualization

To provide network operators with an intuitive understanding of network incidents, we have developed a visualization frontend for SkyNet, as depicted in Figure 11. In this graph, nodes symbolize network devices, while edges represent the connections between these devices. The temporal and spatial scope of the graph is delimited by incidents identified by SkyNet. Devices and links are highlighted based on the outcomes of alert voting; an alert generated by a device or link registers a vote for itself and the connected links or devices. In severe incidents, the visualization frontend aids network operators in swiftly localizing the root cause. For example, during a logic-site incident last year involving multiple devices at the logic-site level, the visualization frontend identified the device with the highest voted score as a reflector, which is not a common logic-site level device. As a result, the failure was promptly mitigated, effectively reducing its duration.

7.2 Heuristic Rules

Before implementing SkyNet, we deployed a network diagnosis and mitigation system based on heuristic rules manually formulated from historical failures. The following example illustrates the rule operation:

- If a device within a group is detected to be losing packets.
- If other devices within this group do not generate alerts.
- If the total traffic through this group is below a specified threshold.

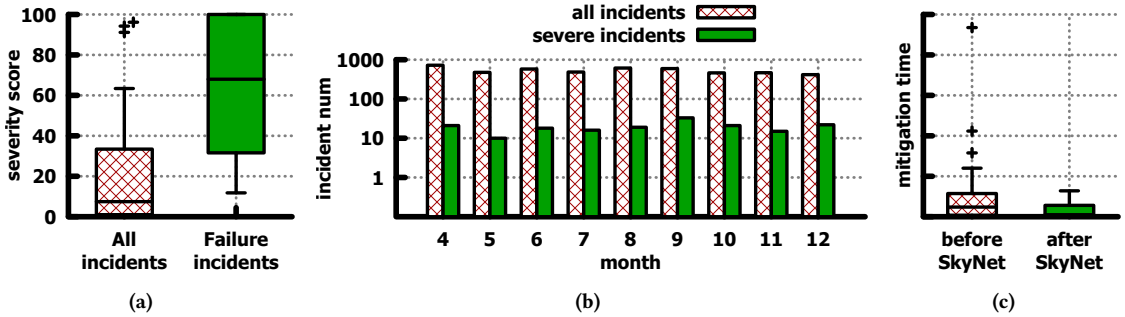

Figure 10: (a) Severity score of network incidents. (b) Incident number before and after filter counted by month. (c) Comparison of mitigation time before and after deploying SkyNet.

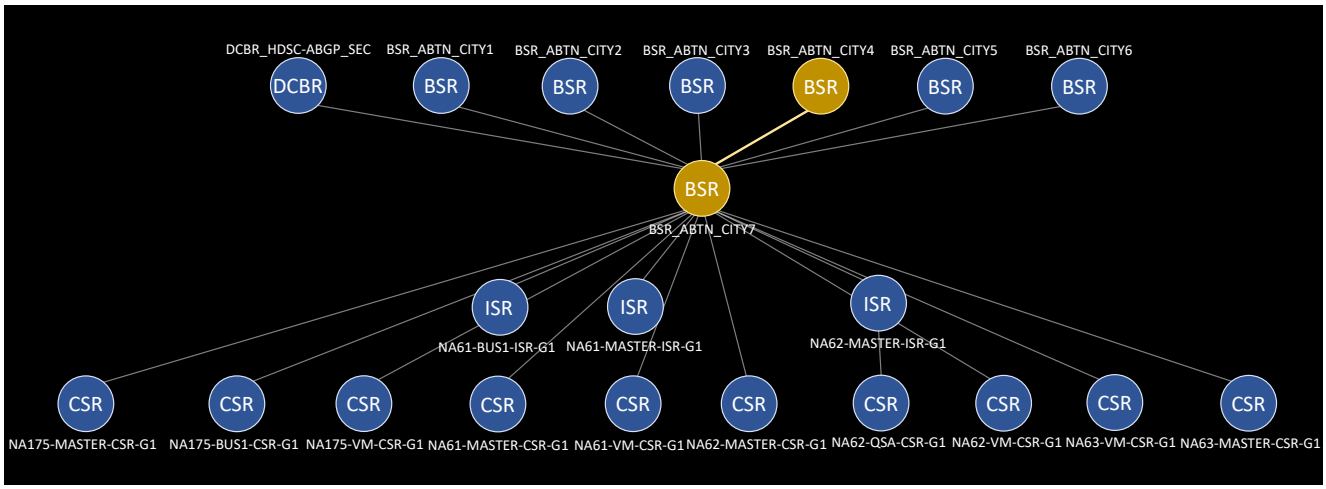

Figure 11: Visualization of SkyNet.

When these conditions are met, the monitoring system diagnoses a device-induced packet loss and devises a plan to isolate the faulty device. If any conditions are unmet, mitigation is not initiated. Concurrently, a rollback plan is prepared, enabling network operators to manually revert actions to prevent incorrect mitigation. Despite the operators having developed nearly 1,000 rules, it remains challenging to address every network failure with these rules alone. For instance, last year we encountered a failure in which all entry links to a data center were broken. This situation was unprecedented, thus no heuristic rule could effectively address it. However, SkyNet rapidly identified that the alerts were concentrated on a single data center, enabling resolution within 15 minutes. Currently, our network employs both heuristic rules and SkyNet, with minor network issues typically resolved by the heuristic system and severe failures addressed by SkyNet.

7.3 Time-series relationship among alerts

In common sense, time series analysis is employed to establish causal relationships between alerts, where the first alert is seen as the root cause, and subsequent alerts are considered effects. However, this approach is not always reliable in practice. Usually network behavior is affected first, then the logs of root cause alerts are collected, such as device errors or interface failures. In an incident last year, both an unbalanced hash and device hardware error jointly caused a network failure. Contrary to expectations, the device hardware error was not the initial alert; instead, a BGP link break alert was the first to occur, followed by an flood of packet drop and device unreachable alerts. Several minutes later, SkyNet received a syslog indicating the device had encountered a hardware error, which is finally proved to be the root cause of the incident. Based on these observations, we choose not to use time series to decide the relationship between alerts, but use a alert tree with time-out window to associate alerts and detect incidents.

8 Related Work

Network monitoring tools. To do fast network failure mitigation, we should start from network monitoring tools [3, 6, 10, 15, 17, 18, 27, 35–38, 42, 50, 60]. We define the network monitoring tools here as the works using various techniques to get aware of the status of the network. The outputs are information possibly related to network failures, for example, packet-level or flow-level reachability issues, device level hardware or software errors. Some of these works [7, 19, 24, 34, 55] have the ability of zooming the root cause

of failure into a specific link or device, and even mitigate it, but as we mentioned before, because they only depends on single data source, they may have limitation of coverage. On the other hand, in the view of SkyNet's workflow, the output of them is the input of SkyNet's preprocessor.

Meanwhile, there are other works focusing on optimizing the existing network monitoring tools to improve their performance, granularity, reliability and availability [4, 8, 13, 14, 16, 21–23, 26, 30, 31, 41, 45, 52, 53, 58, 59]. Although they does not import new data sources to SkyNet, the improvement of data quality also improves the accuracy of SkyNet.

Network diagnosis systems. Not all of them directly collect information from the network, but they analyze the information collected by network monitoring tools [12, 28, 29, 40, 46, 49], and trying to point out the root cause of network failures. Other tools [33, 48, 61] recommends mitigation plans with least impact to network services in the given range of some heuristic operations. Due to the limitation of data sources or heuristic diagnosis rules, they cannot meet our demand on coverage or cannot solve the alert flooding issue. The difference and relationship between SkyNet and LLM-based solutions [5, 44, 47] has already been discussed in §2.3.

Incident prioritizing. There are many existing works [25, 32, 43, 51, 57] that evaluate the severity and priority of bugs in general programs and systems. However, there are few works specifically aims at evaluating the severity of network failures. DeepIP [9] use the history data to train a model with deep learning to evaluate the severity of network incidents. But for severe network failures it is impossible to get enough history data for model training therefore we can only use heuristic way like in SkyNet.

9 Conclusion and Future Work

This paper presents SkyNet, a practical network monitoring system with high coverage of network failures by importing various network monitoring data sources. SkyNet avoid alert flooding by clustering network alerts into incidents, filtering network failures which are not urgent. SkyNet has been stably running in our network over the past nine months, reducing the network failure mitigation time by over 80%. We are actively working on importing more data sources into SkyNet to further improve its coverage and reliability, and adjusting parameters to reduce false positives.

Our future work would focus on the following directions.

More data sources. To enhance the coverage of SkyNet, we are currently integrating additional network monitoring data sources, such as `user-side telemetry`, which transmits telemetry packets from users' clients to the data center. For our newly designed SRTE network, we are utilizing a label-based testing tool to periodically verify link reachability. After being structured, the alerts raised by these tools can be simply injected into SkyNet.

Better thresholds. SkyNet employs several experience-based thresholds, including those for alert numbers in incident generation and severity scores in evaluation. In the future, with the accumulation of more experiential data, it will be possible to implement AI solutions, such as deep learning, to develop more precise and dynamically adaptive thresholds.

Integration with LLM. The use of SkyNet is not inconsistent with failure localization tools that rely on large language models (LLMs). On the contrary, the time and location data extracted from incidents identified by SkyNet can serve as valuable inputs for LLMs. In theory, SkyNet truncates the monitoring results to maintain compliance with the LLM input length constraints without sacrificing valuable information. We are currently exploring the integration of SkyNet and LLMs to enhance the precision of root cause localization in network failures.

Acknowledgements

We acknowledge all teams within Alibaba Cloud that contributed to the success of SkyNet, including the Network Automation, Network Operation, and Network Systems, to name a few. We thank our shepherd, Ryan Beckett, and the SIGCOMM reviewers for their insightful comments. Ennan Zhai and Tao Lin are the corresponding authors.

This work does not raise any ethical issues.

References

[1] 2024. Annual Outage Analysis 2024 - Uptime Institude. https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2024.

[2] 2025. Alibaba Cloud's Global Infrastructure. https://www.alibabacloud.com/en/global-locations.

[3] Aijay Adams, Petr Lapukhov, and J Hongyi Zeng. 2016. NetNORAD: Troubleshooting networks via end-to-end probing. *Facebook White Paper* (2016).

[4] Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. 2022. {HeteroSketch}: Coordinating network-wide monitoring in heterogeneous and dynamic networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 719–741.

[5] Toufique Ahmed, Supriyo GHOSH, Chetan Bansal, Tom Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE 2023)*. https://www.microsoft.com/en-us/research/publication/recommending-root-cause-and-mitigation-steps-for-cloud-incidents-using-large-language-models/

[6] Ghazanfar Ali, Jon R. Hass, Alan Sill, Elham Hojati, Tommy Dang, and Yong Chen. 2022. Redfish-Nagios: A Scalable Out-of-Band Data Center Monitoring Framework Based on Redfish Telemetry Model. In *SNTA@HPDC 2022, Fifth International Workshop on Systems and Network Telemetry and Analytics, Minneapolis, MN, USA, 30 June 2022*, Massimo Cafaro, Jerry Chou, Jinoh Kim, and Alex Sim (Eds.). ACM, 3–11. doi:10.1145/3526064.3534108

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

[8] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. 2020. PINT: Probabilistic in-band network telemetry. In *Proceedings of the ACM SIGCOMM 2020 Conference*. 662–680.

[9] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. 2020. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 373–384.

[10] Rui Ding, Xunpeng Liu, Shibo Yang, Qun Huang, Baoshu Xie, Ronghua Sun, Zhi Zhang, and Bolong Cui. 2024. RD-Probe: Scalable Monitoring With Sufficient Coverage In Complex Datacenter Networks. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 258–273.

[11] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753* (2024).

[12] Jiaqi Gao, Nofel Yaseen, Robert MacDavid, Felipe Vieira Frujeri, Vincent Liu, Ricardo Bianchini, Ramaswamy Aditya, Xiaohang Wang, Henry Lee, David Maltz, et al. 2020. Scouts: Improving the diagnosis process through domain-customized incident routing. In *Proceedings of the ACM SIGCOMM 2020 Conference*. 253–269.

[13] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. 2019. {SIMON}: A simple and scalable method for sensing, inference and measurement in data center networks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 549–564.

[14] Mojgan Ghasemi, Theophilus Benson, and Jennifer Rexford. 2017. Dapper: Data plane performance diagnosis of tcp. In *Proceedings of the Symposium on SDN Research*. 61–74.

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

[16] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: Query-driven streaming network telemetry. In *Proceedings of the ACM SIGCOMM 2018 Conference*. 357–371.

[17] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown. 2014. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 71–85.

[18] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. 2018. Network-wide heavy hitter detection with commodity switches. In *Proceedings of the Symposium on SDN Research*. 1–7.

[19] Vipul Harsh, Tong Meng, Kapil Agrawal, and Philip Brighten Godfrey. 2023. Flock: Accurate network fault localization at scale. *Proceedings of the ACM on Networking* 1, CoNEXT1 (2023), 1–22.

[20] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2024. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* (Nov. 2024). doi:10.1145/3703155 Just Accepted.

[21] Qun Huang, Patrick PC Lee, and Yungang Bao. 2018. Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference. In *Proceedings of the ACM SIGCOMM 2018 Conference*. 576–590.

[22] Qun Huang, Siyuan Sheng, Xiang Chen, Yungang Bao, Rui Zhang, Yanwei Xu, and Gong Zhang. 2021. Toward {Nearly-Zero-Error} sketching via compressive sensing. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 1027–1044.

[23] Qun Huang, Haifeng Sun, Patrick PC Lee, Wei Bai, Feng Zhu, and Yungang Bao. 2020. Omnimon: Re-architecting network telemetry with resource efficiency and full accuracy. In *Proceedings of the ACM SIGCOMM 2020 Conference*. 404–421.

[24] Chenhao Jia, Tian Pan, Zizheng Bian, Xingchen Lin, Enge Song, Cheng Xu, Tao Huang, and Yunjie Liu. 2020. 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*. IEEE, 1–9.

[25] Ahmed Lamkanfi, Serge Demeyer, Emanuel Giger, and Bart Goethals. 2010. Predicting the severity of a reported bug. In *2010 7th IEEE working conference on mining software repositories (MSR 2010)*. IEEE, 1–10.

[26] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. 2016. {FlowRadar}: A better {NetFlow} for data centers. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*. 311–324.

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

[28] Yuxing Li, Hu Zheng, Chengqiang Huang, Ke Pei, Jinghui Li, and Longbo Huang. 2020. Terminator: An Efficient and Light-weight Fault Localization Framework. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 580–585.

[29] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, et al. 2020. Gandalf: An intelligent,{End-To-End} analytics service for safe deployment in {Large-Scale} cloud infrastructure. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 389–402.

[30] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. 2019. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM SIGCOMM 2019 Conference*. 334–350.

[31] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 101–114.

[32] Tim Menzies and Andrian Marcus. 2008. Automated severity assessment of software defect reports. In *2008 IEEE International Conference on Software Maintenance*. IEEE, 346–355.

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

[34] Yanghua Peng, Ji Yang, Chuan Wu, Chuanxiong Guo, Chengchen Hu, and Zongpeng Li. 2017. {deTector}: a topology-aware monitoring system for data center networks. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 55–68.

[35] Diana Andreea Popescu and Andrew W Moore. 2017. PTPmesh: Data center network latency measurements using PTP. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 73–79.

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

[37] Kwang Sik Shin, Jin Ha Jung, Jin Young Cheon, and Sang Bang Choi. 2007. Real-time network monitoring scheme based on SNMP for dynamic information. *Journal of network and computer applications* 30, 1 (2007), 331–353.

[38] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, Shan Muthukrishnan, and Jennifer Rexford. 2017. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*. 164–176.

[39] William Stallings. 1993. *SNMP, SNMPv2, and CMIP: The practical guide to network management*. Addison-Wesley Longman Publishing Co., Inc.

[40] Malgorzata Steinder and Adarshpal S. Sethi. 2004. Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM transactions on networking* 12, 5 (2004), 809–822.

[41] Haifeng Sun, Jiaheng Li, Jintao He, Jie Gui, and Qun Huang. 2023. OmniWindow: A General and Efficient Window Mechanism Framework for Network Telemetry. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 867–880. doi:10.1145/3603269.3604847

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

[43] Yuan Tian, David Lo, Xin Xia, and Chengnian Sun. 2015. Automated prediction of bug report priority using multi-factor analysis. *Empirical Software Engineering* 20 (2015), 1354–1383.

[44] Haopei Wang, Anubhavnidhi Abhashkumar, Changyu Lin, Tianrong Zhang, Xiaoming Gu, Ning Ma, Chang Wu, Songlin Liu, Wei Zhou, Yongbin Dong, et al. 2024. {NetAssistant}: Dialogue Based Network Diagnosis in Data Center Networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 2024)*. 2011–2024.

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

[46] Zhe Wang, Huanwu Hu, Linghe Kong, Xinlei Kang, Qiao Xiang, Jingxuan Li, Yang Lu, Zhuo Song, Peihao Yang, Jiejian Wu, et al. 2024. Diagnosing Application-network Anomalies for Millions of {IPs} in Production Clouds. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 885–899.

[47] Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. 2024. NetLLM: Adapting Large Language Models for Networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*. https://api.semanticscholar.org/CorpusID:267413129

[48] Xin Wu, Daniel Turner, Chao-Chih Chen, David A Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. 2012. NetPilot: Automating datacenter network failure mitigation. In *Proceedings of the ACM SIGCOMM 2012 Conference*. 419–430.

[49] Yang Wu, Ang Chen, and Linh Thi Xuan Phan. 2019. Zeno: Diagnosing performance problems with temporal provenance. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 395–420.

[50] Kenji Yamanishi and Yuko Maruyama. 2005. Dynamic syslog mining for network failure monitoring. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 499–508.

[51] Cheng-Zen Yang, Chun-Chi Hou, Wei-Chen Kao, and Xiang Chen. 2012. An empirical study on improving severity prediction of defect reports using feature selection. In *2012 19th Asia-Pacific Software Engineering Conference*, Vol. 1. IEEE, 240–249.

[52] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *Proceedings of the 2018 ACM SIGCOMM Conference*. 561–575.

[53] Minlan Yu, Lavanya Jose, and Rui Miao. 2013. Software {Defined}{Traffic} Measurement with {OpenSketch}. In *10th USENIX symposium on networked systems design and implementation (NSDI 13)*. 29–42.

[54] Zhaoyang Yu, Minghua Ma, Chaoyun Zhang, Si Qin, Yu Kang, Chetan Bansal, Saravan Rajmohan, Yingnong Dang, Changhua Pei, Dan Pei, et al. 2024. Monitorassistant: Simplifying cloud service monitoring via large language models. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 38–49.

[55] Hongyi Zeng, Ratul Mahajan, Nick McKeown, George Varghese, Lihua Yuan, and Ming Zhang. 2015. Measuring and troubleshooting large operational multipath networks with gray box testing. *Mountain Safety Res., Seattle, WA, USA, Rep. MSR-TR-2015-55* (2015).

[56] Shenglin Zhang, Weibin Meng, Jiahao Bu, Sen Yang, Ying Liu, Dan Pei, Jun Xu, Yu Chen, Hui Dong, Xianping Qu, and Lei Song. 2017. Syslog processing for

switch failure diagnosis and prediction in datacenter networks. *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)* (2017), 1–10. https://api.semanticscholar.org/CorpusID:26472285

[57] Tao Zhang, Jiachi Chen, Geunseok Yang, Byungjeong Lee, and Xiapu Luo. 2016. Towards more accurate severity prediction and fixer recommendation of software bugs. *Journal of Systems and Software* 117 (2016), 166–184.

[58] Hao Zheng, Chengyuan Huang, Xiangyu Han, Jiaqi Zheng, Xiaoliang Wang, Chen Tian, Wanchun Dou, and Guihai Chen. 2024. μMon: Empowering Microsecond-level Network Monitoring with Wavelets. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 274–290.

[59] Yang Zhou, Ying Zhang, Minlan Yu, Guangyu Wang, Dexter Cao, Eric Sung, and Starsky Wong. 2022. Evolvable network telemetry at facebook. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 961–975.

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

[61] Danyang Zhuo, Monia Ghobadi, Ratul Mahajan, Klaus-Tycho Förster, Arvind Krishnamurthy, and Thomas Anderson. 2017. Understanding and mitigating packet corruption in data center networks. In *Proceedings of the ACM SIGCOMM 2017 Conference*. 362–375.