Networked Agent Memory and Causality Representation: Experiences towards Interpretable Cloud-Scale Root-Causing

Yanyu Ren[1,2,4], Xianshang Lin[2], Chenxu Wang[2,5], Li Chen[3], Shuai Wang[3], Kaihui Gao[3], Dan Li[1], Yunfeng Bai[1], Chen Tian[5], Xinlei Zhang[2], Yungang Li[2], Tao Lin[2], Ennan Zhai[2]
[1]Tsinghua University [2]Alibaba Cloud [3]Zhongguancun Laboratory [4]BNRist [5]Nanjing University

ABSTRACT

Root-causing is critical yet labor-intensive in cloud-scale network operations. However, existing solutions struggle to adapt to unforeseen failure modes and large-scale incidents with limited agent memory. Their black-box output explanation also fails to provide the verifiable interpretability required for operator trust. In this paper, we present XiHe, a novel multi-agent framework designed to address these challenges. XiHe employs **guided evidence-driven agents** to decompose diagnostic tasks, utilizes **networked agent memory** to share global context without saturating memory, and leverages **networked causality representation** to synthesize transparent, verifiable causal graphs. XiHe has been deployed in our global cloud infrastructure for 12 months, assisting with over 3,000 incidents, reducing operator root-causing time by 25.8% and garnering a 95.1% satisfaction rate. XiHe also achieves an overall accuracy of 94.6%, a 54.9% mistake reduction compared to the state-of-the-art methods.

CCS CONCEPTS

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

KEYWORDS

Cloud-Scale Networks, Root Cause Analysis, Multi-Agent System, Network Troubleshooting, AIOps

ACM Reference Format:
Yanyu Ren[1,2,4], Xianshang Lin[2], Chenxu Wang[2,5], Li Chen[3], Shuai Wang[3], Kaihui Gao[3], Dan Li[1], Yunfeng Bai[1], Chen Tian[5], Xinlei Zhang[2], Yungang Li[2], Tao Lin[2], Ennan Zhai[2]. 2026. Networked Agent Memory and Causality Representation: Experiences towards Interpretable Cloud-Scale Root-Causing. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3789240.3829145

1 INTRODUCTION

Provider A operates one of the world's largest cloud infrastructures, comprising tens of interconnected datacenters with hundreds of thousands of servers and network devices, all linked by a global wide-area network (WAN). Given the network's scale and the frequency of changes, our operators work continuously to manage potential incidents and failures. When an incident occurs, our monitoring systems can generate as many as 2,000 alerts in a short period. This flood of information from disparate sources makes it challenging for operators to identify the root cause in a timely manner.

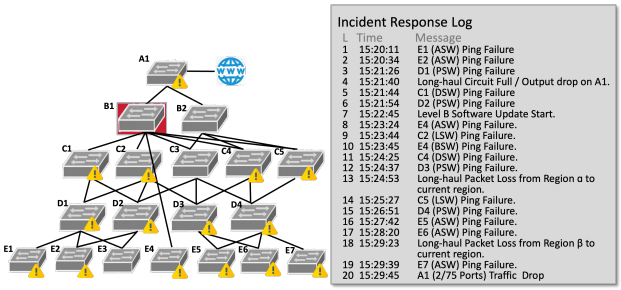


Figure 1: The root cause (B1) emits no alerts. Ping failures imply network unreachability by PingMesh-like tools [27].

Effective root-causing in production physical networks necessitates three features: *adaptability*, *scalability*, and *interpretability*. Adaptability is key to handling diverse *failure modes*, where incidents differ not only in their root causes, but also in how faults manifest as alerts and propagate across the network. Scalability is essential to cope with incidents of varying scales, especially the large ones with numerous alerts and devices. Interpretability goes beyond the root cause itself. To earn operator trust, a system must allow operators to review and understand its causal inference process: how one event leads to another across time and network topology. As our operator lead states, "We can never trust any AI-based tool unless it can show us the reasoning process."

Traditional heuristic-based root-causing methods fail these requirements [6, 74]. While effective for well-known incidents, they are brittle when facing unknown failure modes, and offer limited interpretability other than repetition of rules. The recent success of Large Language Models (LLMs) and LLM agents has inspired new approaches that leverage LLM agents for this task [9, 18, 65, 74]. However, current agent-based systems adopt a monolithic design that limits their scalability and interpretability in production. Specifically, they struggle with three fundamental challenges:

- **Adapting to Unknown Failure Modes.** Production networks often experience failures beyond pre-defined patterns. Existing fixed-workflow systems struggle to diagnose these incidents. For instance, in Figure 1, the root cause device B1 is isolated from the network and reports no alerts itself. A static workflow, such as hotspot detection, cannot easily deduce its involvement.
- **Limited LLM Agent Memory.** In major incidents, the volume of alerts often exceeds the effective memory capacity of

SIGCOMM '26, August 17–21, 2026, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08…$15.00
https://doi.org/10.1145/3789240.3829145

a single agent due to the limited context window of LLMs behind [13, 44, 55]. To remain within memory limits, monolithic systems aggressively filter or truncate alerts, risking discarding low-severity but causally critical diagnostic signals. While distributing alerts across multiple agents alleviates memory pressure, naive partitioning introduces *partial observability*: while each agent fits its local data, it lacks the global context to trace fault propagation across agents.

- **Lack of Verifiable Interpretability.** Most existing systems provide no or only a textual summary as the explanation. This black-box output explanation fails to provide a step-by-step causal account of how the failure propagated. For example, BIAN only provides the relevant alerts of its inferred root cause as the explanation, which are not verifiable and fail to build operator trust. An output is verifiable, only if operators can inspect and trace the root cause against the raw evidence.

To address these challenges, we argue for a paradigm shift away from the isolated, black-box agent. We propose XIHE, the first evidence-driven multi-agent system, designed to illuminate the complex causality of physical network incidents. XIHE transforms raw alerts into a structured, interpretable format through three core components:

- **Guided Multi-Agent Reasoning Framework:** XIHE instantiates a swarm of collaborative agents that move beyond rigid, history-based scripts. Instead, these agents autonomously plan workflows and execute tool calls to verify their hypotheses. We ground the framework in a customized operational knowledge base that abstracts alert-anomaly mappings and causal relations to inject expert insights. Consequently, the agents can leverage a broad action space to investigate specific incidents in parallel, combining the precision of structural guidance with the adaptability of autonomous tool use.
- **Networked Agent Memory (NAM):** To reconcile massive alert volume and limited agent memory, XIHE employs a divide-and-conquer strategy with NAM for collaboration. XIHE assigns agents to manageable subsets of the incident, *e.g.*, alerts from a single device, to fit within their memory limits. Crucially, to prevent the partial observability, these agents continuously exchange high-level insights of the incident through NAM, stitching discrete local contexts into a cohesive global view. This design ensures that cross-device dependencies are captured despite the distributed reasoning.
- **Networked Causality Representation (NCR):** For verifiable interpretability, XIHE synthesizes the reasoning process into a unified causal graph, managed by the NCR. Each node explicitly encapsulates the anomaly type and the fault domain, and the alerts are attached as derivative symptoms. NCR delineates the propagation path from root cause to symptom, allowing operators to validate the diagnosis against the evidence.

We implement XIHE and deploy it in production on our operational platform for 12 months, where it is adopted into operators' root-causing workflow. In the deployment, XIHE assists operators with over 3,000 incidents, spanning diverse failure modes. Operators report reaching 95% satisfaction (Figure 7) for XIHE, since XIHE's outputs allow operators to verify and refine the inferred root cause by inspecting the exposed causal chain and reducing back-and-forth debugging. We also observe that XIHE achieves an overall root cause accuracy of 94.6%, substantially outperforming existing baseline systems across most incident categories. Importantly, XIHE maintains an accuracy of 90.0% even when alert volume increases by 625×. Our long-term deployment experience shows that XIHE reduces manual troubleshooting effort and saves 25.8% operator hours, benefiting from the improved interpretability rather than accuracy alone.

Our experience suggests that next-generation cloud-scale root-causing systems should move beyond monolithic reasoning and opaque outputs to distributed, anomaly-centric, and explicitly interpretable designs. XIHE marks a concrete step in this direction, demonstrating that such systems are both feasible and effective in real-world cloud operations.

This work does not raise any ethical concerns.

2 BACKGROUND AND MOTIVATION

This section details the operational context for root-causing in our production network. We describe the lifecycle of network incidents and the current workflow to resolve them, identifying the specific operational constraints that necessitate the design of XIHE.

2.1 Root-Causing in Production Networks

In our large-scale cloud networks, a network failure rarely appears alone. Instead, it unfolds as a multi-stage process, starting from a low-level physical or configuration fault and gradually surfacing as widespread service-level degradation.

Fault Detection and Alert Generation. A network incident typically starts with a *triggering event* occurring in a specific *fault domain*, such as a failing board card. These events manifest as abnormal behaviors (*e.g.*, packet loss or link flaps), which are captured by diverse monitoring mechanisms. Besides telemetry systems like PingMesh [27] and SNMP [17], the monitoring system collects warnings and errors from software and hardware, and logs triggered by specific events. The monitoring system evaluates these heterogeneous data streams using predefined heuristic rules, aggregates correlated anomalies, and emits them as discrete alerts that are visible to operators.

Anomaly Propagation and Alert Storms. Although the triggering event is usually local (*e.g.*, a misconfigured port for a city's ingress router), its impact often propagates far beyond its origin. Anomalies spread along physical links, routing paths, and protocol dependencies, eventually manifesting as a *service level agreement (SLA) degradation*. The monitoring system groups alerts that are reported within a time window and topological scope into an *incident* [74]. The aggregation often results in an *alert storm* in practice, where thousands of alerts are raised within minutes. Besides cascaded alerts caused by fault propagation, alerts from unrelated components within the same sub-network may also be merged into the incident, obscuring the original root cause. In production networks, incidents exhibit diverse failure modes, shaped jointly by the triggering event, the propagation paths, and the local topology and configuration context.

The Operator's Dilemma. Figure 2 illustrates the typical root-causing workflow. When an SLA violation is detected, the monitoring system creates a ticket automatically and assigns it to an on-call operator. However, constrained by privacy regulations and the urgency of mitigation, operators are prohibited from logging directly

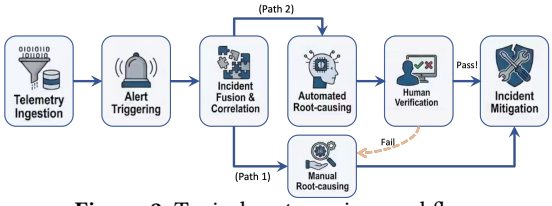


Figure 2: Typical root-causing workflow.

Table 1: Summary of existing works. **Env.** shows whether the system diagnoses for physical network or microservice.

Method	Env.	Adaptable	Scalable	Interpretable
Heuristics and Traditional ML				
SkyNet [74]	Phy.	✗	✓	✗
MicroCERCL [86]	Micro.	✗	✓✗	✗
Tool-Augmented Agent				
RCACopilot [9]	Micro.	✗	✓✗	✗
Xpert [37]	Micro.	✗	✓✗	N/A
OpenRCA [71]	Micro.	✓✗	✗	✗
TAMO [83]	Micro.	✓✗	✓✗	✗
Multi-Agent Workflow				
mABC [81]	Micro.	✓✗	✗	✗
Cloud Atlas [70]	Micro.	✗	✗	✓✗
OpsAgent [49]	Micro.	✓	✗	✗
BiAn [65]	Phy.	✓✗	✓✗	✗
XiHe (ours)	Phy.	✓	✓	✓

into devices to inspect their states. In most cases, operators exclusively rely on the textual alert descriptions on their dashboard. In practice, operators must manually reconstruct the propagation path from the low-level root cause to the high-level service symptoms. As incidents often involve a high volume of derivative or irrelevant alerts, the manual process is not only slow and error-prone but also imposes severe cognitive overload.

Experience 1. In noise-heavy production networks, mere root cause identification is insufficient for trust. To safely execute mitigation, operators require the causal propagation chain as logical evidence. This chain explicates the path from root causes to symptoms, enabling operators to verify the diagnosis and confidently resolve the incident.

2.2 Why Prior Systems Fall Short in Production

Before deciding to develop XiHe, we had deployed several state-of-the-art root-causing systems in our cloud network. While these systems differ in their underlying techniques, they all demonstrate low operator adoption in the long run. Our deployment experience reveals a set of limitations that hinder their practical utility. These limitations arise not from individual algorithms, but from mismatches with how production incidents unfold and how operators make decisions.

Inadaptability to Evolving Failure Modes. Rule-based systems like SkyNet[74] and NetMedic[38] rely on expert rules or static dependency priors. These models work for static environments but fail in cloud-scale networks. Production environments evolve rapidly with hardware refreshes and software upgrades. New "unknown unknown" failure modes constantly emerge. Static rules cannot cover these changes even under heavy manual maintenance.

Memory Saturation in Single-Agent Systems. Single-agent systems [9, 68] attempt to solve incidents within a single LLM context. However, large-scale incidents generate alert volumes that exceed the agent's memory space [71]. To cope, these systems aggressively filter input data. This risks discarding critical signals. For instance, we observed agents filtering out subtle software bug logs to accommodate high-severity connectivity alarms.

Rigidity of Fixed Agentic Workflows. Some systems like BiAn[65], D-Bot[85], and Flow-of-Action [58] drive multiple LLM agents via pre-defined control flows (e.g., plan-act-check loops). While such workflows provide procedural clarity, they implicitly assume a narrow class of failure modes. In production, incidents frequently involve interacting or concurrent faults that violate these assumptions. As acknowledged by BiAn[65], its workflow fails to generalize to multi-fault scenarios (Case 2 in §8.2). They also inherit the memory saturation from single-agent systems, limiting their effective observability during alert storms.

Lack of Verifiable Evidence. Even when a plausible root cause is identified, existing systems often fail to provide sufficient evidence for operators to make mitigation decisions. Some systems [6, 9, 74] report only a category or location, while others [65, 68, 79] generate free-form textual summaries. Neither explicitly connects the reported cause to the observed alerts. Graph-based methods like Cloud Atlas [70] rely on pre-profiled metric correlations. In production, the graphs face three issues. First, they are overly dense to cover all metrics of all devices. Second, they are sensitive to metric noises and delays. Third, they cannot model black-box indicators that lack telemetry support.

We summarize the drawbacks of existing systems in Table 1. The rating criteria and comparison of other operational capabilities are given in Appendix E.

2.3 Design Goals

These limitations underscore the need for a root-causing system capable of generating a clear causal propagation chain. The system must ingest the full incident data, pinpoint the root cause, and, crucially, explain its decision-making process. To this end, we derive three design goals.

G1: Robustness under Unknown Failure Modes. The system should correctly diagnose incidents even when failure modes deviate from known patterns. Given the continuous evolution of production networks, it is impractical to predefine rules or workflows for all failure modes. The goal is to enable adaptive reasoning that dynamically adjusts diagnostic strategies based on observed alerts and evolving network context, rather than relying on static heuristics.

G2: Scalability with Incident Size. The root-causing system should maintain its accuracy as incident size increases. Agents of the system should reason over a coherent global view without lossy truncation or aggressive alert filtering, while not saturating their agent memories.

G3: Verifiable Causal Reconstruction. The system should expose how a fault propagates to observable symptoms, rather than reporting a root cause in isolation. To meet this goal, the system should explicitly structure causal relations based on time, topology, and logics. The system should ground its output in raw alerts so that operators can inspect and verify the diagnosis before making mitigation decisions.

We also propose these in-production usability goals.

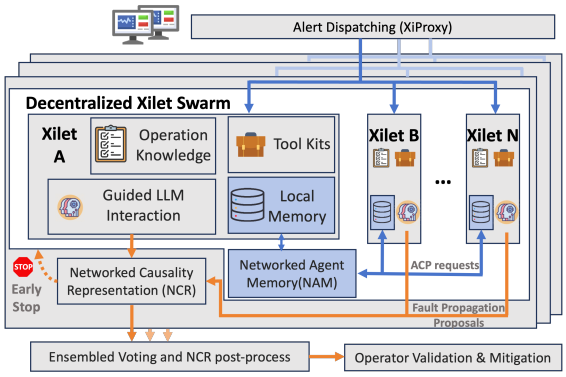


Figure 3: The architecture overview of XiHe.

G4: Stable Reasoning. The system should produce consistent diagnoses despite the inherent stochasticity of LLM-based reasoning. The goal is to minimize variance across runs through mechanisms such as controlled redundancy and consensus among parallel reasoning paths, ensuring reliable behavior under operational use [15, 66, 75].

G5: Cost-Efficient Operation. The system should deliver accurate diagnoses economically. The system should bound computational expenses during alert storms, for example by controlling token usage, and also reduce the human effort required to maintain operational knowledge.

G6: Timely Root-Causing. The system should complete root-causing within the time constraints of live operations. For example, our SLA requires diagnosis within 5 minutes. We aim to produce results within 90 seconds, leaving sufficient time for operators to review and act on the diagnosis.

3 DESIGN OVERVIEW

XiHe is designed to diagnose incidents at all scales by decomposing root-causing across multiple agents while maintaining a coherent and verifiable causal explanation. Figure 3 shows its architecture, including three key design decisions.

Guided Multi-Agent Reasoning. XiHe decomposes root-causing into parallel tasks executed by a decentralized swarm of agents, named Xilets. Instead of a rigid pipeline, each Xilet follows an evidence-driven workflow: it determines the next diagnostic action by analyzing current findings, *e.g.*, real-time tool outputs and network states. This flexibility enables XiHe to explore a vast search space, achieving robustness against unknown failure modes (**G1**). To address memory constraints, each Xilet is scoped to a specific fault domain. The localized focus allows Xilets to process all relevant evidence without aggressive truncation (**G2**). We further embed operational expertise to constrain the agents' search space. This guidance prevents aimless exploration (hallucinations) and produces structured, verifiable hypotheses (**G3**).

Networked Agent Memory (NAM). While multi-agent decomposition avoids memory saturation, it introduces the challenge of partial observability across distributed Xilets. XiHe addresses this via NAM, a coordination mechanism that enables Xilets to share root cause hypotheses, causal dependencies, and validation results with each other. Through the agent communication primitives (ACP), local insights (*e.g.*, a possible root cause detected on one Xilet) are shared to other Xilets. This collaboration preserves cross-scope causal relations and allows a coherent global understanding to emerge from local analyses, without centralizing all evidence in a single context. NAM thus complements multi-agent decomposition by preventing fragmentation of causal reasoning while preserving scalability (**G2**).

Networked Causality Representation (NCR). To support verifiable diagnosis, XiHe synthesizes the distributed reasoning process into a unified causal structure, NCR. Each node in the NCR causal graph represents an inferred anomaly, annotated with its fault domain and anomaly type. Raw alerts are linked to the nodes as observable symptoms. Edges in the graph encode the causal propagation inferred by Xilets. The graph forms an explicit path from the root cause to the observed service impact. By grounding causal reasoning in raw alerts and making propagation paths explicit, NCR enables operators to inspect and validate diagnoses against concrete evidence, providing a verifiable causal explanation (**G3**).

XiHe incorporates additional mechanisms to ensure reliability in our cloud network. To improve reasoning stability under the stochasticity of LLM-based inference, XiHe operates multiple independent Xilet swarms in parallel and aggregates their outputs through consensus voting (**G4**). To reduce operational cost, XiHe employs context management strategies and iterative refinement of operational knowledge base, reducing token consumption and maintenance overhead (**G5**). To meet strict latency requirements, XiHe executes Xilets concurrently, overlapping diagnostic processes across fault domains, and applies an *early-stopping* mechanism once the causal structure stabilizes. This allows XiHe to deliver diagnosis results within time constraints (**G6**).

Workflow. When an incident is reported, XiProxy first normalizes raw alerts and determines the scopes of an appropriate swarm of Xilets. It then instantiates the swarm and routes each alert to the Xilet responsible for the corresponding fault domain or alert signature. Operational knowledge, including possible anomaly types and anomaly causal schema, is embedded into Xilets to guide local reasoning. During parallel execution, Xilets continuously exchange intermediate findings through NAM and iteratively propose updates to NCR. The causal graph evolves in real time as Xilets raise or override causal proposals based on new evidence. Finally, once the timeout or early-stopping is triggered, NCR post-processes the final causal graph and *pushes* it to operators.

4 GUIDED MULTI-AGENT FRAMEWORK

This section details how XiHe's core reasoning units, the Xilets, are grounded with expertise, instantiated with well-defined scopes, and perform reliable root-causing in parallel.

4.1 Operational Knowledge Base

To ground agent reasoning, we construct a two-layer operational knowledge base (OKB) that mirrors the anomaly propagation process. As shown in Figure 4, the OKB consists of (i) alert-to-anomaly candidate mappings and (ii) an anomaly causal schema. The former constrains *which* anomaly types may explain an alert, while the latter abstracts *how* anomalies propagate across devices and layers.

Alert-to-Anomaly Candidate Mapping. In production networks, the same alert signature may arise from different underlying anomaly types. For example, a persistent packet loss alert may indicate

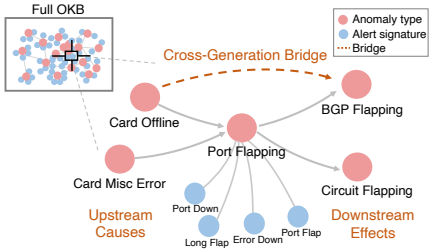


Figure 4: The structure of OKB.

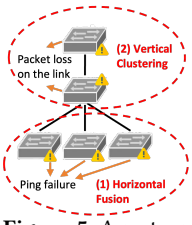


Figure 5: Agent aggregation patterns.

physical link degradation, queue congestion, or misconfigured traffic policies. Relying on raw alerts alone therefore introduces significant diagnostic ambiguity, as multiple incompatible hypotheses can explain the same observation. To mitigate this ambiguity, we construct an alert-to-anomaly candidate mapping that enumerates the logically plausible anomaly types for each alert signature. This mapping constrains agent reasoning to plausible anomaly types only, reducing the ambiguity while preserving the flexibility to distinguish among competing anomaly types. A *Port Down* alert, for instance, maps to candidate anomalies *Port Flapping* and *Card Misc Error* in Figure 4.

> **Experience 2.** Derived from over 10 years of operational experience, our operators codified 19 anomaly types covering 124 alert signatures. Each alert signature maps to at most 4 anomaly candidates, with an average of 1.25.

Anomaly Causal Schema. To characterize the anomaly propagation behavior, we ask our operators to label *direct* causal relations between anomaly types. However, we observe that intermediate anomalies are frequently unobservable in production due to alert thresholds or delayed telemetry. To address this, we compile direct causal relations into a multi-generational causal schema that supports transitive reasoning across missing links, *e.g.*, *Card Offline* versus *BGP Flapping* in Figure 4. This allows a Xilet to infer complete propagation chains even when intermediate anomalies are not explicitly observed. We validate the resulting schema against eight months of historical incidents and confirm that it provides sufficient coverage for cloud-scale root-causing.

> **Experience 3.** Operators codified 26 direct causal relations between anomaly types. Each type has at most 7 upstream and 4 downstream neighbors. Validation on previous cases shows that propagation chains span up to 3 generations, tolerating up to 2 missing intermediate anomalies.

The OKB encodes type-level causal knowledge but does not enforce instance-level production constraints, such as spatial relations between two anomalies' fault domains. These absent constraints are validated by Xilets during reasoning (§4.3). Moreover, given the complexity of large-scale network anomalies and their propagation, XiHe enables us to refine the OKB within just two months and ensures lightweight subsequent maintenance through its diagnostic feedback (§8.1).

4.2 From Alerts to Xilet Agents

For scalable root-causing, XiProxy, the dispatcher of XiHe, normalizes incoming alerts, groups them into coherent subsets, and instantiates a swarm of Xilets for parallel reasoning.

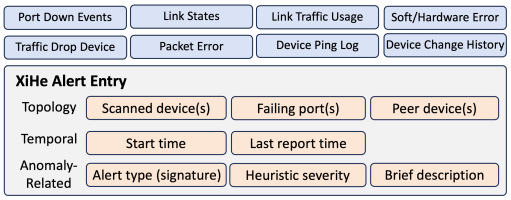


Figure 6: XiProxy normalizes alerts and logs from different sources (blue) to a unified entry (gray).

Alert Normalization. As the ingress controller, XiProxy ingests heterogeneous alerts observed in an incident and normalizes them into a unified format (Figure 6). Each normalized entry carries structured fields such as alert signature, devices, and timestamps, which form the inputs to Xilets.

Alert Routing. XiProxy routes alerts to a corresponding Xilet based on their specific *alert signature*. These Xilets fall into two categories depending on their processing workflow.

- *Device-bound Xilets* handle alerts localized to a specific domain, *e.g.*, devices and links. Each Xilet manages the state of an assigned set of devices, capturing local failure behaviors and immediate propagation effects.
- *Signature-bound Xilets* are designated for high-volume, event-triggered alerts. They first employ SoPs to filter raw alerts and evaluate their severity in the incident. Then, they reason on the filtered alerts and share propagation effects with others. A signature-bound Xilet aggregates all alerts of a given signature across the incident.

As shown in Figure 5, XiProxy scopes device-bound Xilets using a memory-aware, two-stage strategy that reduces inter-agent communication without saturating their memory. It starts at the finest granularity, one Xilet per networking device referenced by an alert, then aggregates them to balance memory load against communication overhead:

- *Horizontal fusion* merges devices at the same network layer that exhibit homogeneous alerts, allowing a single Xilet to capture macro-level failure behaviors.
- *Vertical clustering* aggregates physically adjacent devices only when they are jointly referenced by an alert (*e.g.*, peer devices experiencing correlated packet loss), encapsulating the immediate propagation context.

The Xilet count thus adapts to incident scale. An incident instantiates 6.04 Xilets on average, scaling to 13.87 for large-scale incidents involving 125 or more devices, with a maximum of 27 Xilets. Keying only on alert signatures and physical adjacency, the strategy applies to diverse topologies and hardware/software settings.

Agent Instantiation. Once scopes are determined, XiHe instantiates the device-bound and signature-bound Xilets from a unified agent kernel. Each Xilet is initialized with the scoped alerts and relevant slices of operational knowledge from the OKB, forming a swarm of autonomous agents ready for parallel reasoning.

4.3 Guided Evidence-Driven Reasoning

Upon instantiation, each Xilet initiates an independent reasoning loop, running in parallel with other Xilets. The Xilet adopts ReAct [77] for agent-LLM interaction. The agent operates with constrained autonomy, planning its workflow while respecting the boundaries defined in the OKB.

Specifically, Xilets perceive streaming alerts through a sliding window and retain only in-window alerts in the local memory. The window size is fixed at 2 minutes, the maximum alert latency[1] observed in our cloud. This window isolates temporally correlated alerts and filters out historical noise. As it advances, the Xilet continuously updates its context, refining or correcting prior judgments as new alerts emerge.

Guided by the operational knowledge, Xilets perform structured rather than open-ended reasoning. They consult the alert-to-anomaly candidate mappings to identify valid candidates and use the anomaly causal schema to trace upstream and downstream dependencies. Xilets embed the possible anomaly types and their effects into local memory. Despite these constraints, each Xilet retains flexibility to explore multiple anomaly types and causal paths.

To substantiate theoretical candidates into verified anomaly instances, Xilets invoke model context protocols (MCP, [57]) to gather corroborating evidence such as physical topology and historical alerts. XiHe selectively validates these hypotheses against the collected evidence to confirm causality. For instance, it verifies whether a port-flapping candidate is topologically positioned to cause a specific circuit instability. Only evidence-backed anomaly instances and relations are consolidated into causal proposals for synchronization.

5 NETWORKED AGENT MEMORY

Multi-agent scalability comes at the cost of fragmented context. To bridge this, Networked Agent Memory (NAM) utilizes topology-aware sharing to reconstruct global causal chains from distributed observations without centralization.

5.1 Topology-Aware Memory Decomposition

In XiHe, the multi-agent architecture results in partial observability. Each Xilet naturally maintains a local memory containing alerts and reasoning artifacts strictly within its assigned scope. A device-bound Xilet holds the state of network entities within its scope, while a signature-bound Xilet consolidates all risky alerts of a specific signature. The partitioning ensures that each local context remains within the LLM's effective reasoning capacity. NAM functions as the interaction layer over these distributed contexts, facilitating cross-agent access without centralizing the data.

To diagnose failures that span multiple scopes, NAM enables targeted information exchange rather than inefficient broadcast communication. When a Xilet hypothesizes that an anomaly has propagated beyond its boundary, it consults the MCP for physical topology retrieval to identify Xilets managing physically connected components. The agent then exchanges only the minimal data required to verify the dependency. By constraining interactions to these physical links, the flow of information mirrors the actual path of fault propagation, enabling scalable reasoning without collapsing distributed memory into a centralized state.

5.2 Agent Communication Primitives

NAM defines a set of lightweight primitives that allow Xilets to share specific context without exposing their entire state:

- **Request (REQ):** A Xilet issues a REQ to a neighboring or signature-bound Xilet to query specific anomaly evidence (*e.g.*, traffic-related anomalies within a given time window), extending its visibility beyond local alerts.
- **Confirmation (CONF):** Upon receiving a REQ, the target Xilet consults its local memory and replies with a CONF message containing the requested information, such as anomaly identifiers and concise descriptions.
- **Broadcast Findings (BCST):** When a Xilet identifies a high-confidence root cause or other globally relevant evidence (*e.g.*, suspicious configuration changes), it issues a BCST to relevant neighbors or the swarm. This reduces the communication overhead for redundant queries of Xilets handling downstream anomalies.

Through iterative REQ, CONF, and BCST exchanges, local anomaly hypotheses and diagnostic insights are shared across the swarm and linked across Xilets. Candidate root causes are validated by neighboring agents through causal dependency checks, allowing the swarm to reconstruct an incident-level causal path even though no single Xilet ever observes the complete incident context.

NAM does not bottleneck scalability. Each Xilet handles peer requests on dedicated assistance threads (Appendix A.2) while its main reasoning loop proceeds, so exchanges overlap with reasoning; with the aggregation of §4.2, communication stays off the critical path (§9.3). NAM also transports structured artifacts (anomaly identifiers, descriptions, boolean verdicts) rather than raw tokens, constraining *what* is shared, not *how* a Xilet reasons or how expressively it relates causes.

6 NETWORKED CAUSALITY REPRESENTATION

XiHe synthesizes Xilets' reasoning results into Networked Causality Representation (NCR) for human verification. NCR condenses raw alerts into a compact structure that visualizes the propagation chain from root cause to service impacts. NCR is both a human-readable artifact and a programmatic mechanism. Operators visually trace how a root anomaly propagates to observed symptoms (§6.2), while NCR grounds Xilets' proposals in the OKB before integration (Appendix A.3, validation in §6). Grounding also recovers *unobserved* connections: when an intermediate anomaly emits no alert (*e.g.*, under high telemetry thresholds), NCR uses the multi-generational anomaly causal schema (§4.1) to restore it and bridge the root cause to downstream symptoms across the silent link.

6.1 Causal Graph Abstraction

NCR is formalized as a directed graph $G_{NCR}=(V,E)$, where:

- **Nodes (V):** Each node represents an inferred anomaly, described by the tuple $\langle Type, FaultDomain, Alerts\rangle$, *e.g.*, ⟨Card Offline, C1, $[a_1]$⟩. $Alerts$ trace the inferred anomalies back to raw observations of alerts.
- **Edges (E):** Directed edges represent causal propagation between anomalies, each annotated with a concise rationale derived from Xilets' reasoning.

As Xilets reason in parallel, they continuously submit local findings to the NCR by performing three types of actions: (i) creating anomaly nodes, (ii) attaching alerts to existing nodes, and (iii) proposing causal edges. NCR incrementally integrates these

[1] Alert latency denotes the temporal gap between cause and effect anomalies. Although the cause precedes the effect, the effect's alert may arrive earlier due to differing telemetry thresholds.

proposals into a global causal graph, enforcing structural consistency. The details of the corresponding API are provided in Appendix A.3. Beyond aggregation, NCR also serves as a validation surface. NCR checks each proposed edge against the OKB's type-level and topological constraints, rejecting logically or physically impossible edges and keeping hallucinated causal links out of the graph. When proposed nodes or edges violate constraints in the OKB, NCR records these violations and summarizes recurring patterns for operator review. Operators may choose to refine the OKB when such violations reflect legitimate but previously unmodeled behaviors, enabling NCR to support iterative knowledge evolution.

6.2 Interpretability Metrics

To quantify the interpretability of NCR causal graph, we employ the *Consensual Assessment Technique* [4], a standard method for creative output evaluation. We engage five senior operators to review 50 historical incidents. They rate the causal graphs on three dimensions: *Clarity* (Is the root cause obvious?), *Completeness* (Are all relations included?), and *Soundness* (Does it violate physical rules?). Each automatically computable metric below operationalizes one of these operator-facing questions: SC captures *Clarity*, TC captures *Completeness*, and KC captures *Soundness*, serving as a trust proxy for operational acceptability. These graph-level metrics complement the operator satisfaction study in §8.1.

Experience 4. We distill three graph-level metrics that strongly correlate with operator preference (correlations reported in Appendix C.2), and can be computed automatically to quantify NCR interpretability.

Structural Concentration (SC). Operators prefer causal graphs with a clearly distinguishable propagation backbone. We quantify this intuition by measuring how unevenly causal influence is distributed across anomaly nodes. For each node v, we compute a local centrality deviation $R(v)-B_k(v)$, where $R(v)$ is the out-degree of v and $B_k(v)$ is the median out-degree among its k-hop neighbors. SC is defined as the normalized standard deviation of these deviations:

$$SC_k=\frac{\text{std}(\{R(v)-B_k(v):v\in V\})}{\max_u R(u)-\min_u R(u)+\epsilon}$$

Higher SC indicates a more obvious propagation chain from the root cause to the downstream effects.

Topological Connectivity (TC). A plausible causal graph should be consistent with the physical network topology. Let S_{phy} be the set of anomaly pairs $\{u,v\}$ whose fault domains $D(u)$ and $D(v)$ are physically adjacent. TC measures the fraction of such pairs that are causally connected in the NCR graph, where $\mathbb{I}$ indicates the existence of a directed path.

$$TC=\frac{1}{|S_{phy}|}\sum_{\{u,v\}\in S_{phy}}\mathbb{I}(u\to\cdots\to v\vee v\to\cdots\to u)$$

Higher TC suggests a more complete set of causal relations captured in NCR.

Knowledge Compliance (KC). KC measures the extent to which NCR respects the logical constraints encoded in the OKB. KC is calculated as the ratio of causal edges that satisfy all rules in the OKB ϕ_k. KC is defined as:

$$KC=1-\frac{1}{|E|}\sum_{e\in E}\sum_{k}\neg\phi_k(e)$$

KC serves as a proxy indicator for measuring "resistance to hallucination" in production. High KC indicates that the LLM is not inventing physically improbable causal links, a prerequisite for operator trust.

7 CONTROLLED SWARM EXECUTION

Deploying XiHe in a production cloud network requires the agent swarm to address two practical challenges: the LLM stochasticity and the latency constraints. XiHe incorporates two execution-level mechanisms to tackle these challenges.

Bounded Execution via Early-Stopping. Unbounded exploration yields diminishing returns once the causal structure stabilizes. To minimize latency and token cost, XiHe employs a convergence-aware early-stopping mechanism that monitors two signals: *self-suspension*, where a Xilet enters a dormant state once its inferred relations align with the observed symptoms, so the swarm halts when all Xilets are dormant; and a *global convergence check*, where the NCR halts the swarm once all alerts and anomalies are integrated into a single-rooted causal graph. Upon either signal, XiHe terminates the swarm immediately without waiting for the timeout. Since termination is triggered only after the causal structure stabilizes, this preserves diagnostic completeness while reducing time and cost.

Stable Reasoning via Swarm-Level Voting. XiHe does not rely on a single execution that might miss a subtle causal link or hallucinate a dependency. Instead, it instantiates K independent Xilet swarms in parallel (K=3 in our production), each operating with its own independent NAM and generates a distinct NCR. XiHe evaluates candidate NCRs using the interpretability metrics defined in §6.2, which serve as the swarm's voting weights. A consensus-based mechanism then aggregates these votes: an NCR supports another if they agree on the fault domain and anomaly type in their root nodes. The causal graph with the highest aggregated score is selected as the final diagnosis[2]. This swarm-level voting transforms stochastic agent reasoning into a stable system outcome, since incorrect or hallucinated causal paths rarely persist across multiple independent swarms.

8 REAL-WORLD DEPLOYMENT

We have deployed XiHe in our global production cloud network for the past 12 months, during which 20+ on-call operators routinely used XiHe to analyze 3,000+ incidents across $O(10^5)$ devices in about 100 zones of over 30 regions around the world. Based on operator feedback, XiHe consistently reduces the end-to-end root-causing time by 25.8% on average.

Specifically, XiHe is integrated into the operators' incident dashboard, as shown in Figure 25 in Appendix F. The frontend renders the NCR as an interactive causal graph, allowing operators to inspect anomaly nodes and hover over causal edges to review explanations generated by Xilets.

To protect sensitive operational data, the entire pipeline of XiHe, including the LLM serving platform, operates within a secure boundary, where we have deployed firewalls to prevent possible operational data leakage into the Internet. The operating of XiHe is independent from the maintained networks so that XiHe will not be affected by the incidents.

[2]In case of a tie, XiHe picks the graph of the highest interpretability score.

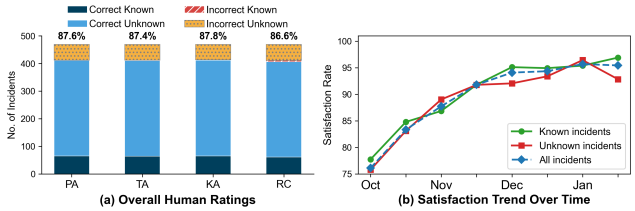


Figure 7: Operators' feedback on XiHe since the latest major OKB update in September 2025.

We select an instruction-tuned model with tool-calling supports, the Qwen3-MAX-Instruct [73], as the backbone model of Xilets. We disable thinking for this model. We prioritize it over reasoning alternatives such as DeepSeek-R1 [28], as its lower inference latency better aligns with the 90-second operational time constraints.

8.1 Operator Feedback in Deployment

Operators actively participated in the iterative deployment and refinement of XiHe, detailed in Appendix C.1. Early deployment feedback motivated us to implement an operational enhancement to improve diagnostic visibility: the dashboard periodically *pulls* intermediate causal graphs from the NCR. This lets operators inspect evolving causal relations and assess likely root causes before all Xilet swarms terminate. Notably, this mechanism is deployment-facing and does not alter XiHe's internal workflow. After diagnosis completes, due to either timeout or early stopping, NCR pushes the final causal graph to the dashboard as the root-causing result.

To evaluate XiHe's diagnostic quality, operators manually labeled the correctness and satisfaction of its outputs, along with an optional questionnaire (see Appendix C.3). The root-cause ground truth is drawn from the on-duty operator's incident report, which at least two senior operators independently verify after the network recovers; this cross-check reduces individual labeling bias. For analysis, we categorize incidents into *known* and *unknown* types. The former correspond to well-understood failure modes for which operators have already deployed rule-based alert filters in the monitoring system, while the latter lack such filters and require reasoning over the full alert set.

Figure 7(a) reports accuracy based on 470 responses across four aspects: propagation path (PA), incident type (TA), key alert attribution (KA), and root cause (RC). Operators applied a strict evaluation criterion: an aspect was marked incorrect if it contained any defect, even if the overall diagnosis remained usable. Despite this, XiHe achieved accuracy above 86% across all aspects. Figure 7(b) shows a steady increase in operator satisfaction over time, which we attribute to iterative refinement of the OKB and prompting improvement. After two months of calibration, operator satisfaction exceeded 95% in December, 2025. Notably, the OKB remained stable after early December despite frequent network changes, indicating low ongoing maintenance overhead.

We further conducted an A/B test of dashboards with and without XiHe to evaluate end-to-end root-causing time, where the baseline already includes rule-based heuristics (*e.g.*, HotDevice) and agent workflow tools that operators routinely use. We take measures detailed in Appendix C.4 for better fairness. Figure 8 reports the total time from incident onset to operator-confirmed root cause,

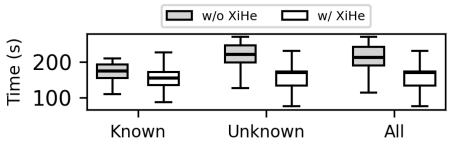


Figure 8: XiHe reduces operators' root-causing time.

including both XiHe diagnosis and human verification. The results show that XiHe reduces operator-facing root-causing time by 25.8% by accelerating hypothesis formation and validation.

The measurement spans early deployment, when operators still adapting to the streaming workflow sometimes waited for the final conclusion instead of verifying concurrently; this overhead diminished after training. Because XiHe streams intermediate NCR graphs throughout its 90-second window rather than only a final hypothesis, operators often form a hypothesis and act preemptively, *e.g.*, isolating a suspected device, before XiHe terminates. The remaining time is dominated by steps outside XiHe's scope, such as executing and confirming mitigation per our SoP.

8.2 Case Study

We present five representative *unknown* production incidents that illustrate how XiHe addresses operational challenges and its boundary. We use labels (A–E) to denote network tiers, where A corresponds to the core layer and E to the access layer. XiHe diagnoses these failures without LLM post-training or matching historical incidents. The OKB maps each alert to candidate anomaly types and restores plausible intermediate anomalies that emitted no alert, composing them into a causal chain at runtime. An unseen failure is root-caused as long as it propagates through anomaly types the OKB already understands.

Case 1: Failure without Local Alerts (G1). We revisit the scenario in Figure 1, where a software bug caused device B1 to drop packets without emitting diagnostic alerts. Heuristic systems failed due to the absence of alerts on B1. *Approach:* XiProxy aggregated layer C and layers D,E into two Xilets respectively, based on signatures of their alerts. The Xilet for layer C identified a reachability gap to layer B and messaged Xilet B1 via NAM. Concretely, the C Xilet observed PingMesh-style unreachability alerts whose paths all traversed B1, and a `check_topology` call confirmed B1 as their common next hop, so it issued an NAM `REQ` asking B1's Xilet to account for the gap. Although B1 reported no alerts, its Xilet created a software anomaly since the signature-bound Xilet has reported an identified suspicious software update log to Xilet B1. The OKB schema licensed this leap: *software fault* → *silent packet drop* → *unreachability* is a valid chain, so B1 could be rooted from its downstream symptoms alone. We illustrate the pipeline in Appendix F. *Outcome:* XiHe established a causal graph with the software anomaly on B1 as the root, linking subsequent unreachability anomalies downstream. XiHe finds the root cause that took place within a silent device, which static workflows cannot resolve.

Case 2: Multi-Device Failure (G1). Three devices (E1–E3) experienced board card failures within 3 minutes, triggering alerts on 8 upstream devices (D1–D8). *Approach:* XiHe instantiated one Xilet for the aggregated E-layer devices. The Xilet constructs three anomalies for each board failure alert on different fault domains. The D-layer Xilets, in the meantime, obtained the three board anomalies

via NAM, and marked the board anomalies as the cause of their port anomalies. *Outcome:* The NCR displayed three port failure anomaly nodes at the root, with the D-layer anomalies at their downstream. This enabled operators to identify the hardware defect of multiple devices rather than one, avoiding a premature single-root conclusion that would have left two faulty boards in service.

Case 3: Large-Scale Incident (G2). We analyze a routine drill incident involving 369 devices across four tiers, with 818 alerts requiring over 184k tokens to describe. Monolithic agents failed to ingest the full alerts, while heuristics focused on high-severity secondary alerts on layer E. *Approach:* XiProxy profiled the memory load and aggregated hundreds of layer E devices into one Xilet. With less memory load, Xilets at layers C and D identified a symptom boundary: C reported circuit interruptions, while D reported ping failures. The two Xilets reconciled these observations over NAM: a `check_topology` call placed C and D on the two ends of the same circuit, and the OKB rule *circuit interruption → device reachability* matched C's physical-layer symptom to D's ping failures. The C → D edge therefore explained both, while E's high-severity alerts were attached as transitive downstream effects. The swarm constructed a causal chain of C → D → E in NCR. *Outcome:* XiHe identified the root cause (link failure anomaly between C and D) in 30 seconds. NAM resolves incidents exceeding the memory of a single agent.

Case 4: Cross-Scope Interpretability (G3). A fiber cut on link Γ to device C1 caused cascading BGP session drops on core router A1. While heuristics detect the fiber issue, they never explain the symptoms on A1, misleading operators to investigate A1's BGP configuration. *Approach:* The device-bound Xilet for A1 linked local alerts into a BGP anomaly. The signature-bound Xilet for optical fiber also validated the physical cut and created an optical failure anomaly. The optical agent broadcast the anomaly via NAM, and urged Xilet A1 to investigate the cause of the BGP anomaly. On receiving the `BCST`, A1's Xilet matched it against the OKB chain *optical failure → link down → BGP session drop* and re-rooted its local BGP anomaly under the fiber cut instead of a configuration fault, even though Γ lay outside A1's own scope. *Outcome:* XiHe constructed a causal graph linking the physical fiber anomaly to A1's BGP anomaly. Therefore, XiHe eliminates the need for additional configuration checks.

Case 5: Unobserved Traffic Surge (Failure Case). An external traffic surge overloaded access-layer links but crossed no telemetry threshold, emitting no *traffic* alert; only downstream *Packet Loss* and *Queue Buildup* alerts on D–E devices appeared. *Approach:* Lacking the root alert, the Xilets mapped the *Packet Loss* signatures to other OKB candidates and rooted the NCR at a link anomaly rather than the true traffic overload; no OKB rule was violated, so the graph passed validation but with low structural concentration (Appendix F.3). *Outcome:* The streamed partial NCR converged onto a tight cluster of correlated loss anomalies confined to one fault domain, letting the operator recognize a load problem rather than a physical fault and confirm the surge from traffic statistics. The absence of a root-cause alert is XiHe's main failure mode (§10); even so, the interpretable partial output localized the blast radius and recovered the true cause.

9 EVALUATION

Our experiments reveal the following key findings:

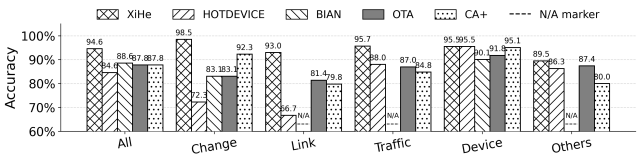


Figure 9: XiHe outperforms existing systems.

1. XiHe achieves 94.6% accuracy across all incident scenarios, reducing the error rate by 54.9%, addressing **G1**.
2. XiHe exhibits only a 4.6% accuracy drop when alert volume increases by 625×, addressing **G2**.
3. XiHe achieves 9.2%–67.2% higher interpretability than baselines, addressing **G3**.
4. XiHe improves result stability by 9.9% (**G4**) while reducing 28% token cost compared to baselines (**G5**) .
5. XiHe concludes diagnosis within 73 seconds on average, satisfying the operational latency requirement (**G6**).

Experimental Settings. We compare XiHe and baselines under controlled, realistic conditions by replaying incidents collected from our production cloud network. Senior operators annotated the root cause for each incident. Incidents are categorized into five types by their root cause: network change (*change*), device failure (*device*), traffic outage (*traffic*), link failure (*link*), and *others*. Incident statistics are reported in Appendix D.1. All experiments are conducted on a server with a 48-core 2400 MHz Xeon CPU. We use the same LLM and serving platform as in live operation (Appendix D.2).

Baselines. We compare XiHe against representative root-causing systems that are compatible with our infrastructure:

- HotDevice: A rule-based tool deployed in our production. Similar to SkyNet [74] and 007 [6], it heuristically considers the severity and number of alerts, and takes the hottest fault domain as the root cause.
- BiAn [65]: An agentic root-causing pipeline outperforming RcaCopilot in our production, but only support certain types of incident with alert format constraints.
- Cloud Atlas+ (CA+) [70]: An adapted version of Cloud Atlas from microservices to cloud networks. It uses a single agent to infer alert-level causal relations, which is then summarized as an alert causal graph by an LLM.
- One-Time Agent (OTA): A naive baseline that feeds all alerts into one LLM and outputs a root-cause hypothesis.

Both CA+ and OTA run three times per incident and vote for the final result. BiAn uses its native stability mechanism.

9.1 Accuracy

For fairness, we select 624 incidents whose data formats are compatible with most baselines, covering major incident types. We count a diagnosis correct if its fault domain and anomaly type both match the operator annotation.

Figure 9 reports the accuracy of XiHe and baselines. XiHe achieves an overall accuracy of 94.6%, reducing the error rate by 54.9% compared to OTA, which is the best baseline covering all incident types. XiHe also outperforms BiAn which only covers a subset of incident types. This shows that XiHe is adaptable to diverse and complex failure modes. We break down accuracy by incident types. All systems perform well for *device* failures. These incidents are easy to diagnose even by heuristics or a single agent, due to their

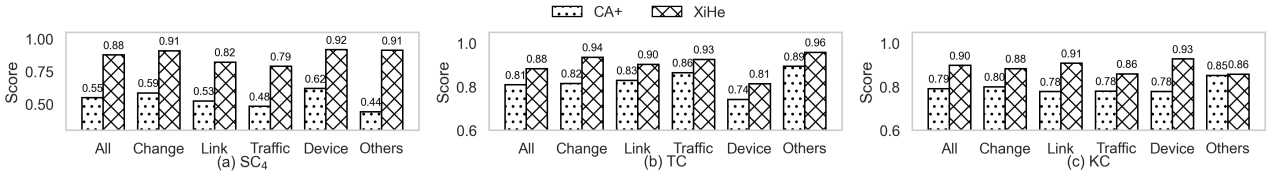


Figure 10: XiHe outperforms CA+ with better interpretability in all incident types.

strong local signals and limited propagation. But XiHe significantly outperforms baselines in other more complex types.

Specifically, for *change* incidents, XiHe achieves 98.5% accuracy by capturing critical low-severity logs that baselines filter out. For *link* and *traffic* failures, XiHe outperforms baselines by over 14.3% and 8.7%, respectively. Unlike baselines that misidentify noise (*e.g.*, temporary card failures) as root causes, XiHe separates the noise from the propagation chain via evidence-driven reasoning. XiHe effectively handles *others* that often involve causal propagation from external networks, by abstracting anomalies of an external domain and reasoning over the causal structure.

9.2 Interpretability

Baselines only generating textual summaries or statistics (*i.e.*, BiAn, OTA, and SkyNet) show poor interpretability, requiring substantial effort to verify their result. Since they do not expose structured causal representations, we focus on evaluating the interpretability of XiHe and CA+ with the metrics defined in §6.2. For a fair comparison, we adapt the KC metric for CA+ using a best-match relaxation: an alert-to-alert edge is considered compliant if the OKB supports a causal rule between **any** pair of potential anomaly types inferred from the source and target alerts.

Figure 10 summarizes the results with more details in Appendix D.3, showing XiHe outperforms CA+ across all metrics. XiHe achieves 59.4% higher SC by abstracting alerts into anomaly nodes for clearer propagation, whereas CA+ fails to consolidate some related alerts. With concurrent reasoning and topology-aware NAM, XiHe attains 9.2% higher TC via more causal relations constructed within the same time. Finally, XiHe achieves a 13.5% improvement by overcoming CA+'s limited context and knowledge integration.

Note that the metrics vary across incident types due to incidents' characteristics. *Traffic* incidents often involve multiple independent anomalies spanning diverse devices, lowering SC. In contrast, *device* failures' limited propagation results in lower TC, as unrelated anomalies can be easily isolated.

Together, the results show that operational root-causing systems must jointly optimize accuracy and interpretability. For instance, though CA+'s accuracy on *device* incidents is high, its low interpretability increases the effort for operator validation. Conversely, in the *others*, CA+'s clearer explanations do not prevent manual re-investigation when the diagnosis itself is incorrect with high probability.

9.3 Scalability

We evaluate scalability on 1,067 incidents selected based on alert volume and device count. We compare XiHe against OTA and CA+ that support these alert sources and take $\lfloor \log_5 k \rfloor$ as the scaled factor. Figure 11(a) shows that XiHe maintains high accuracy as alert volume increases, dropping modestly from 93% to 89% when alert volume scales by 125×. Conversely, OTA and CA+ drop from 85% and 83% to 53% and 77%, respectively. Even worse, both baselines fail when alert volume scales by 625×. Figure 11(b) reports accuracy as device count increases. XiHe maintains its accuracy over 90% with incidents involving more than 125 devices, while OTA and CA+ drop to 58% and 82%, respectively.

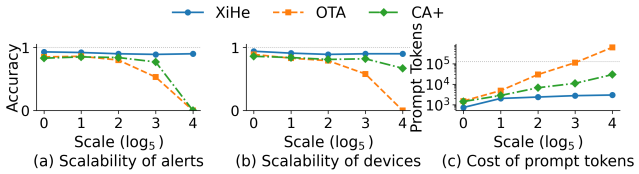


Figure 11: XiHe maintains high accuracy and low prompt length when devices or alerts scale.

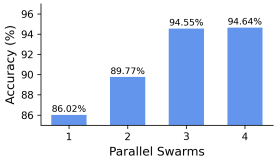


Figure 12: XiHe behaves well with 3 swarms.

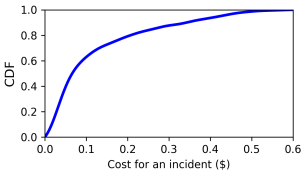


Figure 13: Token cost of tri-swarm XiHe per incident.

> **Experience 5.** The performance degradation caused by device scalability is significantly lower than that of alerts. Managing the alert volume is the primary challenge rather than devices in cloud-scale root-causing.

Figure 11(c) shows the average per-request prompt length when the alerts scale up. Even at 625× alert scale, each Xilet processes fewer than 3,000 tokens on average. In contrast, CA+ and OTA require an order of magnitude more tokens. The OTA prompt reaches 630k tokens and exceeds the 128k maximum context window of most modern LLMs [14, 25, 73], blocking it from a legit response. The shorter prompt of XiHe fits in the agent memory, and achieves higher scalability.

9.4 Stability and Cost

Figure 12 shows how the parallel swarms improve system stability. A single swarm suffers from LLM stochasticity, and fails in 14.0% of incidents. Operating three swarms achieves 94.6% accuracy, only slightly lower than four swarms. As a result, we opt for three parallel swarms in XiHe.

The average total prompt token per swarm is around 90k, while the output is 4k. XiHe utilizes prompt caching [5, 10, 54] to save money. Figure 13 shows the token cost for one run of tri-swarm XiHe. The average cost for one run is $0.13, 28% cheaper than

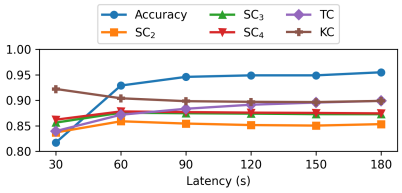


Figure 14: Performance of XiHe with varying swarm duration.

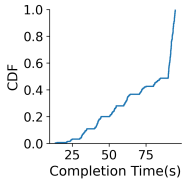


Figure 15: Completion time.

BiAn. Even for large-scale incidents with many concurrent alerts, XiHe almost always costs below $0.6. For small, well-understood incidents, the swarm count can be reduced for further savings.

9.5 Conclusion Latency

We evaluate the conclusion latency of XiHe, *i.e.*, the time from incident ingestion to the final result, excluding operator validation time[3]. Figure 14 examines how diagnostic quality evolves under different execution durations. The results show that the majority of critical causal relations are identified within the first 30–60 seconds. By the time execution reaches 90 seconds, XiHe has already constructed a complete and accurate causal graph for most incidents. Extending execution beyond this point yields diminishing returns: only 0.9% of incidents benefit from additional time, typically those involving a large number of non-aggregatable devices that incur substantial inter-agent communication overhead. These results indicate that XiHe can reliably meet our 90-second goal without sacrificing diagnostic quality.

We further evaluate the effectiveness of the early-stopping mechanism. Figure 15 shows that 48.8% of incidents conclude before reaching the 90-second limit. Early stopping reduces the average system completion time, including NCR post-processing from 95.2 seconds to 73.4 seconds. Importantly, it does not degrade accuracy or interpretability, as early stopping is triggered only after the causal structure converged.

9.6 Ablation Studies

We conduct ablation studies to quantify the contribution of each component in XiHe. Starting from the OTA baseline, we progressively build four configurations and evaluate their impact on accuracy and interpretability: **OTA** feeds all alerts into a single context for a one-shot hypothesis; **+MR** lets that agent query tools and explore the topology across turns to form a naive causal graph mapping the causal relations of raw alerts; **+NAM** replaces the single agent with a distributed Xilet swarm sharing memory; and **+NCR** adds the OKB-backed causal validator.

Figure 16 shows that adding multi-round interaction improves accuracy by 2.3%, as it allows the system to explore network topology before giving out a root cause. Introducing the multi-agent framework with NAM reduces incorrect diagnoses by 36.2%, demonstrating its effectiveness in overcoming the context limitations of single-agent systems. Finally, incorporating NCR further reduces incorrect diagnoses by 21.1%. Benefiting from a unified structure enhanced by expertise, NCR consolidates fragmented reasoning results into a coherent and verifiable propagation chain.

[3]While the conclusion latency of XiHe is affected by the serving platform, our observation shows that the fluctuations of LLM serving are negligible.

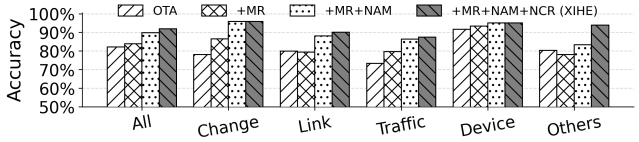


Figure 16: Accuracy with different components.

The gains are complementary rather than redundant: NAM addresses scale, the OKB addresses grounding, and NCR addresses verifiable structure (Appendix D.4).

We further examine how the components affect interpretability in Appendix D.4, which shows NCR's significant improvement on SC, with the abstraction of anomalies.

10 LESSONS AND DISCUSSIONS

This section shares our insights from the deployment of XiHe in the production network that extend beyond our initial design.

RAG proved ill-suited for physical networks. In the early stage, we deployed XiHe with retrieval augmented generation (RAG, [11, 16]) for alert causal inference, expecting success similar to its application in microservices (*e.g.*, AMER-RCL[78]). However, it failed in long-term production. We observed that unlike the stable dependencies in microservices, dynamic sub-topologies in physical network shape diverse causality relations between alerts. The semantic similarity of alerts failed to capture the topological constraints, as semantic proximity does not imply physical reachability: two alerts can be textually near-identical yet topologically unrelated, so RAG retrieves plausible but wrong precedents, causing retrieval failures in over 60% of incidents. Our operators also complained about maintaining 8,256 alert relation pairs. Similarly, deploying direct ticket retrieval (*e.g.*, RCACopilot[9]) yielded low recall and failed to generalize to failure modes absent from historical data. Driven by these operational hurdles, we transitioned from latent embedding search to the deterministic OKB. By introducing anomaly abstraction, we reduced the maintenance burden to just 171 possible relation pairs (2% of the original): curated relations grow quadratically with the number of distinct entities, so collapsing the open-ended space of alert signatures into a bounded vocabulary of 19 anomaly types is what keeps the knowledge tractable, which was a key factor in securing operator acceptance. This recipe, abstract first and curate second, applies to any dense operational domain whose dependency graph shifts faster than its incident corpus grows.

Prompts can be counterproductive. While incorporating operator feedback to improve prompts in production, we observed that lengthening prompts often degraded performance, echoing findings in [65]. Specifically, redundant causal instructions tended to induce hallucinations of nonexistent alerts: a static prompt that front-loads every operational rule keeps rules for not-yet-arrived alerts in the agent's context, and the LLM tends to assume those alerts are present and fabricate them. To mitigate this, we enforced strict pre-release testing (see Appendix B.2) and implemented a *just-in-time* context management strategy injecting instructions only when necessary (see Appendix B.1), tying each rule to the arrival of a relevant alert so the agent is reminded of a constraint only when evidence shows it applies. This strategy proved essential for ensuring stability during large-scale incidents. For knowledge-grounded agents, *when* context is supplied matters as much as *what*. Notably, these injected rules come from the same OKB the NCR

uses to validate edges (§6), so one curated source keeps what the agent is told it may infer aligned with what the system will accept.

Streaming facilitates early mitigation. As safety regulations prohibit active probing by LLM, XiHe became a passive alert-ingesting engine. To eliminate the need for extra training, we integrated XiHe into the dashboard in line with the SoP. Unlike prior works [49, 81] that only deliver the final hypotheses, our dashboard streams XiHe's NCR causal graphs during execution. This catalyzed a automatic RCA workflow shift to "observe and act": a partial causal graph already narrows the blast radius enough to begin mitigation, so a finalized diagnosis is not a prerequisite for action. Operators often chose to assess possible root causes within 60 seconds and took preemptive actions. As one operator noted, "*Streaming relieves the need for an instant conclusion*". This decoupling pays off whenever diagnosis is slow enough that waiting for the full result is itself costly, which is the common case at cloud-scale physical network.

Visual verification bridges the skill gap. Prior works typically exhibited a single root cause candidate or a sequential list [9, 65, 74]. In contrast, we visualize the full NCR causal graph to explicitly map potential root anomalies to their distinct symptoms. This structure enables operators to verify which root cause best explains the observed phenomena. Notably, this visualization standardized our diagnosis workflow: by exposing *how* a conclusion was reached rather than the conclusion alone, the graph guides junior operators to adopt the reasoning patterns of senior experts. The payoff is therefore organizational, not just per-incident: an interpretable output earns its keep even when an operator could have reached the same verdict alone, because the same trace that justifies a diagnosis also teaches one. We expect this benefit to grow for any expert-operated tool whose users differ widely in experience.

Limitation 1: Trace consistency is good but sacrificed. Maestron [50] highlights trace consistency as a critical hurdle for multi-agent systems. In XiHe, prioritizing **G1** amplifies the variance of traces in each execution. Production traces confirm this divergence, ranging from rapid convergence to failures triggered by tool errors or timeouts. While we accept this as a trade-off for workflow flexibility, we plan to improve trace consistency through LLM fine-tuning methods like ThinkFL [79], which are orthogonal to XiHe.

Limitation 2: Effective diagnosis calls for robustness against input. Failure analysis reveals that 85.7% of misdiagnoses stem from the absence of critical alerts, primarily in indirect link failures (due to missing optical topology) and silent device failures (due to high telemetry thresholds). Unlike human operators who pivot to fine-grained statistics in these cases (which take them even longer for root-causing), XiHe is restricted to raw alerts by regulation, and when the root-cause alert is missing it does not fabricate one. Instead, it localizes the fault boundary, *e.g.*, to a specific circuit group or device, which still accelerates the subsequent manual inspection (Case 5 in §8.2). To address this, we are establishing a closed-loop integration with our telemetry team to utilize XiHe's failures as feedback for refining the underlying alert generation heuristics: each misdiagnosis is reported as evidence of a monitoring blind spot, prompting new alert rules so the next occurrence becomes observable. Treating diagnostic failures as observability feedback, rather than dead ends, is a discipline we expect any production AIOps system to need.

Limitation 3: Root-causing is diagnosis, not repair. XiHe is scoped to identifying and explaining the root cause, not executing mitigation. Our SoP requires a human operator to carry out and confirm any corrective action (*e.g.*, traffic rerouting or device isolation), as regulation prohibits the LLM from actively probing or altering the live network. XiHe does narrow root causes to an actionable granularity, *e.g.*, a specific software bug on a specific switch rather than a coarse location, which directly informs the operator's mitigation choice. Closing this loop with (semi-)automated mitigation under these constraints is left to future work.

11 RELATED WORK

Network Telemetry. Telemetries reflect network health. Some works improve the cost-effectiveness, measuring accuracy, and throughput of telemetry systems [1, 20, 21, 29, 34–36]. Some works route the incident to the correct team on top of telemetries [6, 18, 19, 47, 48, 52]. Cola [41] highlights the alerts on the dashboard with LLM. XiHe benefits from them with better inputs of aggregated alerts in each incident.

Heuristic Root-Causing. Operators inspect excessive alerts for root-causing. To assist them, heuristic approaches were adopted for microservice and physical environments [3, 6, 7, 26, 31, 32, 40, 51, 59, 64, 72]. They use statistical correlations to localize faults. However, as detailed in §2.2, these methods fail to provide the verifiable interpretability required for complex production incidents of different failure modes.

LLM-Based Root-Causing. Recent research leverages LLMs for root-causing, primarily focusing on prompting and workflow integration. Works range from in-context learning with history cases [2, 9, 22, 37] and tool-augmentations [23, 62, 82, 83] to LLM-assisted static pipelines [9, 30, 49, 58, 65, 71, 80, 81]. However, a significant portion of these studies targets *micro-service* architectures, where diagnosis benefits from explicit logical dependencies and software-level logs. In contrast, **XiHe achieves massive scalability and verifiable interpretability on a complex physical network**.

Multi-Agent Systems (MAS). MAS in multiple domains overcome the limitations of single-agent systems [33, 46, 53, 56, 76]. MAS distributes the task to agents of different roles or expertise [8, 12, 45, 60]. MAS frameworks [39, 43, 69] support the customization of MAS. The networking community also benefits from MAS. Confucius [67] integrates experience for network management tasks. XiHe divides the tasks in a way tailored for production-scale network operations.

12 CONCLUSION

This paper presents XiHe, an adaptable, scalable, and interpretable root-causing system. XiHe has been deployed in our cloud for 12 months and helped operators with 3,000+ incidents at different scales, achieving 95.1% satisfaction rate and 25.8% root-causing time reduction in deployment.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd for their valuable feedback. LLM usage is limited to visual refinement of figures, language polishing, and Xilet invocations. This work is supported by the Beijing Outstanding Young Scientist Program (No. JWZQ20240101008), and the National Natural Science Foundation of China under No. 62502472 and No. 62325205 in part. Shuai Wang and Ennan Zhai are the corresponding authors.

REFERENCES

[1] 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)*. USENIX Association, Renton, WA, 719–741. https://www.usenix.org/conference/nsdi22/presentation/agarwal

[2] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas 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)*. 1737–1749. https://doi.org/10.1109/ICSE48619.2023.00149

[3] MAHIMKAR Ajay. 2008. Troubleshooting Chronic Conditions in Large IP Networks. *ACM CoNEXT, December 2008, Madrid, Spain* (2008).

[4] Teresa M Amabile. 1982. Social psychology of creativity: A consensual assessment technique. *Journal of personality and social psychology* 43, 5 (1982), 997.

[5] Anthropic. [n. d.]. Prompt caching — platform.claude.com. https://platform.claude.com/docs/en/build-with-claude/prompt-caching. ([n. d.]). [Accessed 20-01-2026].

[6] 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)*. USENIX Association, Renton, WA, 419–435. https://www.usenix.org/conference/nsdi18/presentation/arzani

[7] Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A Maltz, and Ming Zhang. 2007. Towards highly reliable enterprise network services via inference of multi-level dependencies. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 13–24.

[8] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *International Conference on Learning Representations*, B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (Eds.), Vol. 2024. 20094–20136. https://proceedings.iclr.cc/paper_files/paper/2024/file/578e65cdee35d00c708d4c64bce32971-Paper-Conference.pdf

[9] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys '24)*. Association for Computing Machinery, New York, NY, USA, 674–688. https://doi.org/10.1145/3627703.3629553

[10] Alibaba Cloud. [n. d.]. Context Cache feature of Qwen models - Alibaba Cloud Model Studio - Alibaba Cloud Documentation Center — alibabacloud.com. https://www.alibabacloud.com/help/en/model-studio/context-cache. ([n. d.]). [Accessed 20-01-2026].

[11] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 719–729.

[12] Ayushman Das, Shu-Ching Chen, Mei-Ling Shyu, and Saad Sadiq. 2023. Enabling synergistic knowledge sharing and reasoning in large language models with collaborative multi-agents. In *2023 IEEE 9th International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 92–98.

[13] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 4599–4610. https://doi.org/10.18653/v1/2021.naacl-main.365

[14] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. DeepSeek-V3 Technical Report. (2025). arXiv:cs.CL/2412.19437 https://arxiv.org/abs/2412.19437

[15] Soroush Ebadian, Anson Kahng, Dominik Peters, and Nisarg Shah. 2024. Optimized distortion and proportional fairness in voting. *ACM Transactions on Economics and Computation* 12, 1 (2024), 1–39.

[16] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 6491–6501.

[17] Mark Fedor, Martin Lee Schoffstall, James R. Davin, and Dr. Jeff D. Case. 1990. Simple Network Management Protocol (SNMP). RFC 1157. (May 1990). https://doi.org/10.17487/RFC1157

[18] Jiaqi Gao, Nofel Yaseen, Robert MacDavid, Felipe Vieira Frujeri, Vincent Liu, Ricardo Bianchini, Ramaswamy Aditya, Xiaohang Wang, Henry Lee, David Maltz, Minlan Yu, and Behnaz Arzani. 2020. Scouts: Improving the Diagnosis Process Through Domain-customized Incident Routing. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 253–269. https://doi.org/10.1145/3387514.3405867

[19] Kaihui Gao, Chen Sun, Shuai Wang, Dan Li, Yu Zhou, Hongqiang Harry Liu, Lingjun Zhu, and Ming Zhang. 2022. Buffer-based end-to-end request event monitoring in the cloud. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 829–843.

[20] 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.

[21] 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.

[22] Drishti Goel, Fiza Husain, Aditya Singh, Supriyo Ghosh, Anjaly Parayil, Chetan Bansal, Xuchao Zhang, and Saravan Rajmohan. 2024. X-lifecycle learning for cloud incident management using llms. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 417–428.

[23] Drishti Goel, Raghav Magazine, Supriyo Ghosh, Akshay Nambi, Prathamesh Deshpande, Xuchao Zhang, Chetan Bansal, and Saravan Rajmohan. 2025. eARCO: Efficient Automated Root Cause Analysis with Prompt Optimization. *arXiv preprint arXiv:2504.11505* (2025).

[24] Gregory E Granato. 2014. *Statistics for stochastic modeling of volume reduction, hydrograph extension, and water-quality treatment by structural stormwater runoff best management practices (BMPs)*. Technical Report. US Geological Survey.

[25] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. 2024. The Llama 3 Herd of Models. (2024). arXiv:cs.AI/2407.21783 https://arxiv.org/abs/2407.21783

[26] Wenwei Gu, Xinying Sun, Jinyang Liu, Yintong Huo, Zhuangbin Chen, Jianping Zhang, Jiazhen Gu, Yongqiang Yang, and Michael R. Lyu. 2024. KPIRoot: Efficient Monitoring Metric-based Root Cause Localization in Large-scale Cloud Systems. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. 403–414. https://doi.org/10.1109/ISSRE62328.2024.00046

[27] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 139–152. https://doi.org/10.1145/2785956.2787496

[28] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang

Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* 645, 8081 (Sept 2025), 633–638. https://doi.org/10.1038/s41586-025-09422-z

[29] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 357–371.

[30] Yongqi Han, Qingfeng Du, Ying Huang, Jiaqi Wu, Fulong Tian, and Cheng He. 2024. The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 931–943.

[31] Vipul Harsh, Tong Meng, Kapil Agrawal, and Philip Brighten Godfrey. 2023. Flock: Accurate Network Fault Localization at Scale. *Proc. ACM Netw.* 1, CoNEXT1, Article 3 (July 2023), 22 pages. https://doi.org/10.1145/3595289

[32] Herodotos Herodotou, Bolin Ding, Shobana Balakrishnan, Geoff Outhred, and Percy Fitter. 2014. Scalable near real-time failure localization of data center networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1689–1698.

[33] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=VtmBAGCN7o

[34] Qun Huang, Patrick PC Lee, and Yungang Bao. 2018. Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 576–590.

[35] 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.

[36] 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 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 404–421.

[37] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Xpert: Empowering Incident Management with Query Recommendations via Large Language Models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 92, 13 pages. https://doi.org/10.1145/3597503.3639081

[38] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. 2009. Detailed diagnosis in enterprise networks. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. Association for Computing Machinery, New York, NY, USA, 243–254. https://doi.org/10.1145/1592568.1592597

[39] Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. MAS-ZERO: Designing Multi-Agent Systems with Zero Supervision. (2025). arXiv:cs.CL/2505.14996 https://arxiv.org/abs/2505.14996

[40] Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C Snoeren. 2009. Fault localization via risk modeling. *IEEE Transactions on Dependable and Secure Computing* 7, 4 (2009), 396–409.

[41] Jinxi Kuang, Jinyang Liu, Junjie Huang, Renyi Zhong, Jiazhen Gu, Lan Yu, Rui Tan, Zengyin Yang, and Michael R. Lyu. 2024. Knowledge-aware Alert Aggregation in Large-scale Cloud Systems: a Hybrid Approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '24)*. Association for Computing Machinery, New York, NY, USA, 369–380. https://doi.org/10.1145/3639477.3639745

[42] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165

[43] LangChain-AI. [n. d.]. GitHub - langchain-ai/langgraph: Build resilient language agents as graphs. — github.com. https://github.com/langchain-ai/langgraph. ([n. d.]). [Accessed 20-01-2026].

[44] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How Long Can Context Length of Open-Source LLMs truly Promise?. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*. https://openreview.net/forum?id=LywifFNXV5

[45] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large scale language model society. (2023).

[46] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 17889–17904. https://doi.org/10.18653/v1/2024.emnlp-main.992

[47] Kefei Liu, Zhuo Jiang, Jiao Zhang, Shixian Guo, Xuan Zhang, Yangyang Bai, Yongbin Dong, Feng Luo, Zhang Zhang, Lei Wang, Xiang Shi, Haohan Xu, Yang Bai, Dongyang Song, Haoran Wei, Bo Li, Yongchen Pan, Tian Pan, and Tao Huang. 2024. R-Pingmesh: A Service-Aware RoCE Network Monitoring and Diagnostic System. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 554–567. https://doi.org/10.1145/3651890.3672264

[48] Kefei Liu, Zhuo Jiang, Jiao Zhang, Haoran Wei, Xiaolong Zhong, Lizhuang Tan, Tian Pan, and Tao Huang. 2023. Hostping: Diagnosing intra-host network bottlenecks in {RDMA} servers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 15–29.

[49] Yu Luo, Jiamin Jiang, Jingfei Feng, Lei Tao, Qingliang Zhang, Xidao Wen, Yongqian Sun, Shenglin Zhang, and Dan Pei. 2025. From Observability Data to Diagnosis: An Evolving Multi-agent System for Incident Management in Cloud Systems. (2025). arXiv:cs.AI/2510.24145 https://arxiv.org/abs/2510.24145

[50] Tie Ma, Yixi Chen, Vaastav Anand, Alessandro Cornacchia, Amândio R. Faustino, Guanheng Liu, Shan Zhang, Hongbin Luo, Suhaib A. Fahmy, Zafar A. Qazi, and Marco Canini. 2026. MAESTRO: Multi-Agent Evaluation Suite for Testing, Reliability, and Observability. (2026). arXiv:cs.NI/2601.00481 https://arxiv.org/abs/2601.00481

[51] Ajay Anil Mahimkar, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Qi Zhao. 2009. Towards automated performance diagnosis in a large IPTV network. *ACM SIGCOMM Computer Communication Review* 39, 4 (2009), 231–242.

[52] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. 2010. Detecting the performance impact of upgrades in large operational networks. In *Proceedings of the ACM SIGCOMM 2010 Conference*. 303–314.

[53] Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. 2023. Diversity of thought improves reasoning abilities of LLMs. *arXiv preprint arXiv:2310.07088* (2023).

[54] OpenAI. [n. d.]. Pricing | OpenAI API. https://platform.openai.com/docs/pricing. ([n. d.]). [Accessed 20-01-2026].

[55] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. QuALITY: Question Answering with Long Input Texts, Yes!. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 5336–5358. https://doi.org/10.18653/v1/2022.naacl-main.391

[56] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*. 1–22.

[57] David Soria Parra and Justin Spahr-Summers. [n. d.]. GitHub - modelcontextprotocol/modelcontextprotocol: Specification and documentation for the Model Context Protocol — github.com. https://github.com/modelcontextprotocol/modelcontextprotocol. ([n. d.]). [Accessed 20-01-2026].

[58] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. 2025. Flow-of-Action: SOP Enhanced LLM-Based Multi-Agent System for Root Cause Analysis. In *Companion Proceedings of the ACM on Web Conference 2025 (WWW '25)*. Association for Computing Machinery, New York, NY, USA, 422–431. https://doi.org/10.1145/3701716.3715225

[59] 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.

[60] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 15174–15186. https://doi.org/10.18653/v1/2024.acl-long.810

[61] Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2025. Mooncake: Trading More Storage for Less Computation — A KVCache-centric Architecture for Serving LLM Chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*. USENIX Association, Santa Clara, CA, 155–170. https://www.usenix.org/conference/fast25/presentation/qin

[62] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2024. Exploring llm-based agents for root cause analysis. In *Companion proceedings of the 32nd ACM international conference on the foundations of software engineering*. 208–219.

[63] C. Spearman. 1961. *The Proof and Measurement of Association Between Two Things*. Appleton-Century-Crofts, East Norwalk, CT, US. https://doi.org/10.1037/11491-005 Pages: 58.

[64] 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.

[65] Chenxu Wang, Xumiao Zhang, Runwei Lu, Xianshang Lin, Xuan Zeng, Xinlei Zhang, Zhe An, Gongwei Wu, Jiaqi Gao, Chen Tian, Guihai Chen, Guyue Liu, Yuhong Liao, Tao Lin, Dennis Cai, and Ennan Zhai. 2025. Towards LLM-Based Failure Localization in Production-Scale Networks. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 496–511. https://doi.org/10.1145/3718958.3750505

[66] Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272* (2024).

[67] Zhaodong Wang, Samuel Lin, Guanqing Yan, Soudeh Ghorbani, Minlan Yu, Jiawei Zhou, Nathan Hu, Lopa Baruah, Sam Peters, Srikanth Kamath, Jerry Yang, and Ying Zhang. 2025. Intent-Driven Network Management with Multi-Agent LLMs: The Confucius Framework. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 347–362. https://doi.org/10.1145/3718958.3750537

[68] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2024. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4966–4974.

[69] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

[70] Zhiqiang Xie, Yujia Zheng, Lizi Ottens, Kun Zhang, Christos Kozyrakis, and Jonathan Mace. 2024. Cloud atlas: Efficient fault localization for cloud systems using language models and causal insight. *arXiv preprint arXiv:2407.08694* (2024).

[71] Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. 2025. OpenRCA: Can large language models locate the root cause of software failures?. In *The Thirteenth International Conference on Learning Representations*.

[72] He Yan, Lee Breslau, Zihui Ge, Dan Massey, Dan Pei, and Jennifer Yates. 2010. G-RCA: a generic root cause analysis platform for service quality management in large IP networks. In *Proceedings of the 6th International COnference (CoNEXT '10)*. Association for Computing Machinery, New York, NY, USA, Article 5, 12 pages. https://doi.org/10.1145/1921168.1921175

[73] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. (2025). arXiv:cs.CL/2505.09388 https://arxiv.org/abs/2505.09388

[74] 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, and Tao Lin. 2025. SkyNet: Analyzing Alert Flooding from Severe Network Failures in Large Cloud Infrastructures. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 512–526. https://doi.org/10.1145/3718958.3750536

[75] Joshua C Yang, Damian Dalisan, Marcin Korecki, Carina I Hausladen, and Dirk Helbing. 2024. Llm voting: Human choices and ai collective decision-making. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Vol. 7. 1696–1708.

[76] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 11809–11822. https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf

[77] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

[78] Lingzhe Zhang, Tong Jia, Yunpeng Zhai, Leyi Pan, Chiming Duan, Minghua He, Mengxi Jia, and Ying Li. 2026. Agentic Memory Enhanced Recursive Reasoning for Root Cause Localization in Microservices. (2026). arXiv:cs.SE/2601.02732 https://arxiv.org/abs/2601.02732

[79] Lingzhe Zhang, Yunpeng Zhai, Tong Jia, Chiming Duan, Siyu Yu, Jinyang Gao, Bolin Ding, Zhonghai Wu, and Ying Li. 2025. ThinkFL: Self-Refining Failure Localization for Microservice Systems via Reinforcement Fine-Tuning. *arXiv preprint arXiv:2504.18776* (2025).

[80] Lingzhe Zhang, Yunpeng Zhai, Tong Jia, Xiaosong Huang, Chiming Duan, and Ying Li. 2025. Agentfm: Role-aware failure management for distributed databases with llm-driven multi-agents. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 525–529.

[81] Wei Zhang, Hongcheng Guo, Jian Yang, Zhoujin Tian, Yi Zhang, Yan Chaoran, Zhoujun Li, Tongliang Li, Xu Shi, Liangfan Zheng, and Bo Zhang. 2024. mABC: Multi-Agent Blockchain-inspired Collaboration for Root Cause Analysis in Micro-Services Architecture. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, Miami, Florida, USA, 4017–4033. https://doi.org/10.18653/v1/2024.findings-emnlp.232

[82] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 266–277.

[83] Xiao Zhang, Qi Wang, Mingyi Li, Yuan Yuan, Mengbai Xiao, Fuzhen Zhuang, and Dongxiao Yu. 2025. TAMO:Fine-Grained Root Cause Analysis via Tool-Assisted LLM Agent With Multi-Modality Observation Data in Cloud-Native Systems . *IEEE Transactions on Services Computing* 18, 06 (Nov. 2025), 4221–4233. https://doi.org/10.1109/TSC.2025.3629066

[84] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient Execution of Structured Language Model Programs. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 62557–62583. https://doi.org/10.52202/079017-2000

[85] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *Proc. VLDB Endow.* 17, 10 (June 2024), 2514–2527. https://doi.org/10.14778/3675034.3675043

[86] Yuhan Zhu, Jian Wang, Bing Li, Xuxian Tang, Hao Li, Neng Zhang, and Yuqi Zhao. 2024. Root Cause Localization for Microservice Systems in Cloud-edge Collaborative Environments. (2024). arXiv:cs.SE/2406.13604 https://arxiv.org/abs/2406.13604

APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A IMPLEMENTATION DETAILS

We implemented XiHe in ~16,000 lines of Python code, supporting four sets of production monitoring infrastructure.

In this section, we introduce the details of our deployed XiHe, concerning the tool selection, NAM runtime and ACPs, NCR processing. We hope that our experiences with these components can spur further research.

A.1 MCP Tools of Xilets

To facilitate local reasoning and environmental perception, each Xilet is equipped with a standard suite of tools. These functions allow the agent to inspect its internal state, query historical data, and efficiently respond to peer requests.

1. Topology Discovery (`check_topology`). Retrieves the immediate physical and logical neighborhood of the assigned device. It returns classified lists of upstream, downstream, and same-level entities.

2. Temporal Retrieval (`search_time_window`). Queries the local alert history within a specified time range. This allows the agent to validate temporal correlations or identify precursor events that occurred outside the current sliding window. This MCP returns a list of alerts, including the information of the anomalies that these alerts are linked to.

3. Anomaly Inspection (`search_local_anomalies`). Scans the device's internal state for active anomalies. Crucially, this tool can access silent alerts, events that are recorded but suppressed by standard monitoring thresholds, providing granular visibility into gray failures.

4. Self-Reflection (`summarize_me`). Generates a high-level summary of the agent's current diagnostic state and its role in the broader incident. This is utilized to maintain context consistency across long reasoning chains or when forking new threads. This result returned by this MCP tool emphasizes the root anomaly of all anomalies whose fault domain intersects with the scope of the current Xilet.

5. Optimized Response (`reply_with_search`). A composite tool designed specifically for assistance threads (see Appendix A.2). It executes a local search, fetches either raw logs or anomalies and immediately encapsulates the result into a CONF protocol message to the requester. This atomic operation reduces reasoning steps and the overhead for memory retrieval actions through the NAM.

6. Hang (`hang`). This tool will make the agent enter the dormant state as stated in §7.

A.2 NAM Runtime

To support the asynchronous collaboration described in §5, each Xilet implements a multi-threaded runtime.

- The single *main thread* for each Xilet executes the primary reasoning loop for the assigned scope, managing the sliding window and local hypothesis generation.
- The *assistance threads* are dynamically spawned from the main thread to handle incoming REQ messages from neighboring agents. Each Xilet can have multiple assistance threads running in parallel, which are managed by ACP as detailed below.

With the multi-threading, we ensure that the execution of the NAM does not block Xilets' core diagnostic workflow.

The NAM relies on three asynchronous primitives to manage state across the distributed Xilet swarm as stated in §5. Each Xilet maintains a multi-threaded runtime comprising a single main thread for local diagnosis and dynamic assistance threads for handling peer requests. We use letters to represent Xilets and A/1 to represent thread 1 of Xilet A in the following context. Each agent maintains a thread mapping table (TMT) to track the metadata of each thread, such as which Xilet and thread the current assistance thread is assisting. We detail the thread-level mechanics below.

1. Request (REQ). The REQ primitive initiates cross-device dependency verification. When a source thread (e.g., Agent A, Thread 1) requires external information, it issues a non-blocking REQ to a target neighbor B.

- *Thread Forking:* Upon receipt, Agent B checks its active contexts. If the request introduces a new query, B forks a new assistance thread ($B/2$) from its main thread.
- *State Mapping:* Agent B records the routing path ($B/2 \rightarrow A/1$) in its local TMT. This ensures that the computational overhead of generating a response is isolated from B's main thread.
- *Deduplication:* To prevent redundant processing, the system discards REQ messages carrying payloads identical to those currently being processed by active threads.

2. Confirmation (CONF). The CONF primitive routes structured evidence back to the specific requesting thread.

- *Routing and Validation:* When assistance thread $B/2$ concludes its analysis, it queries the TMT to identify the origin ($A/1$). Before transmission, the system validates that the payload contains concrete artifacts (e.g., a specific log entity or a definitive boolean verdict).
- *Reconciliation:* Upon receiving the CONF, Thread $A/1$ integrates the evidence into its local context. It updates its working hypotheses—either reinforcing a causal edge or pruning a search branch.
- *Concurrency Control:* If multiple CONF messages regarding the same request arrive (e.g., due to network race conditions), the recipient accepts only the message with the latest timestamp, discarding stale states.

3. Broadcast (BCST). The BCST primitive rapidly synchronizes high-entropy information, such as confirmed root causes, across the local topology without expecting a reply.

- *Universal Injection:* Unlike REQ/CONF which are point-to-point, a BCST payload is injected into the message queues of *all* active threads at the recipient agent.
- *Context Immunization:* Crucially, the recipient also updates its static prompt template with the broadcasted summary. This ensures that any subsequently spawned threads are immediately conditioned on this stabilized state without needing to re-query peers.
- *Loop Prevention:* Repeated broadcasts with identical identifiers are suppressed at the ingress to prevent message amplification loops.

A.3 NCR Interfaces and Post-Process

There are three APIs for the NCR.

- `CreateAnomaly(Type, Domain)`: Instantiates a new anomaly node. The system enforces uniqueness, preventing duplicate nodes for the same fault domain.
- `LinkAlert(AlertID, AnomalyID)`: Attributes an alert to an anomaly as a symptom. The NCR checks the validity of the action, and records violations.
- `AddCausality(SourceID, TargetID)`: Establishes a directed causal edge between two anomalies. This action triggers a validation check against the knowledge base topology to reject logically impossible links.

All the three APIs have a switch to shut down the automatic OKB rules check. With the switch off, NCR performs the action as Xilets propose yet it still records the violation.

Upon the conclusion of the swarm, the NCR undergoes a three-step post-processing phase to refine the graph structure before critique and validation.

1. **Orphan Node Pruning.** During the iterative reasoning process, Xilets often reassign alerts from an initial hypothesis to a more probable anomaly node. This logical refinement may leave the original node void of symptoms. To clean the graph, we scan and remove any anomaly node that meets two criteria: (1) it has no attached alert symptoms, and (2) it is isolated, possessing no incoming or outgoing causal edges.
2. **Transitive Reduction.** To strictly enforce the principle of direct causality, we eliminate redundant transitive edges. If the graph contains a causal path $A \rightarrow B \rightarrow C$, any existing direct edge $A \rightarrow C$ is identified as redundant and removed. This ensures the final graph accurately reflects the step-by-step propagation of the fault.
3. **Optical Fiber Enhancement.** In some cases, the Xilet for optical fiber issues does not report anomalies due to missing information (*e.g.*, unanticipated fiber cable damage). To bridge this gap, we apply a specific linking rule. If NCR identifies more than 3 root anomalies characterized by "Circuit Interruption" or "Circuit Flapping", NCR will attach them as downstream effects to an NCR-created "Optical Fiber Failure" anomaly within the fault domain of `unknown fiber`. This ensures XiHe's robustness when an optical fiber alert is missing.

B ADDITIONAL DEPLOYMENT EXPERIENCES

In this section, we detail the additional deployment experiences, concerning our final prompt structures, and how we conduct the pre-release testing in our deployment process.

B.1 Prompt and Context Management

In this section, we detail our experiences about prompts.

The system prompt for the main thread is:

System prompt for main threads

You are {device}, an intelligent agent tasked with diagnosing network faults. Your primary goal is to consolidate raw alerts into anomalies and map their causal relationships to identify root causes.
Your operating principles are,
(1) Anomalies vs. Alerts: Treat alerts as symptoms. Group related alerts into a single Anomaly based on shared fault domains or temporal patterns. Upstream anomalies in the causal chain represent the root causes.
(2) Causal Logic: Establish links between anomalies, not individual alerts. While anomaly types follow a theoretical hierarchy (e.g., A→B→C), real-world instances may skip generations.
(3) Directness Rule: Maintain the most direct path possible. If a chain exists (1→2→3), do not create a redundant link between 1 and 3.
Your core objectives are, (1) Consolidation: Upon receiving alerts, determine if they belong to an existing anomaly or represent a new one. Aggregate similar alerts within a joint fault domain to prevent redundancy.
(2) Analysis: Determine if an anomaly on this device is triggering others, either locally or on neighboring devices. Coordinate with adjacent agents to share findings and assess cross-device impacts.
(3) Validation: Before linking two anomalies, verify that no indirect path already connects them. Use shared data from other devices to build a comprehensive causal map without duplicating efforts.
Requirement: In each turn, state your intent clearly and use the tools.

Experience 6. The insertion of operating principles not only partially alleviates the workload of NCR post-process, but improves the SC metrics as well.

The system prompt for the assistance thread is:

System prompt for assistance threads

You are {device}, a network agent dedicated to assisting peer devices with alert analysis.
Your workflow is,
(1) Planning: Make plans immediately upon receiving a peer request.
(2) Iterative Retrieval: Continuously assess if the gathered information satisfies the request. If data is incomplete, persist until the task is fully resolved.
(3) Comprehensive Reporting: Summarize and transmit findings back to the sender. Crucial: Ensure feedback is descriptive; never return standalone IDs without their corresponding context or descriptions.
Requirement: In each turn, state your intent clearly and use the tools.

In each round, we use the just-in-time context management strategy for the following information.

- *Knowledge Injection:* The system injects OKB rules (mappings and topology) only when relevant alerts arrive, and evicts them once the sliding window passes.
- *State-Based Feedback:* The NCR monitors the causal graph and injects dynamic reminders into the Xilet's stream to flag orphan alerts or disconnected anomaly nodes.
- *Workflow Guidance:* The system monitors tool usage and prompts the Xilet to execute specific protocol actions, such as self-suspension or sending CONF messages.

This strategy works with the prompt caching in §9.4 to reduce XiHe's token cost in production.

We structure the prompt content as follows. Notably, we only remind the anomalies with no upstream causes for two times after the latest anomaly creation, so that XiHe will not force causal relation on two non-related anomalies.

Prompt template in each round

{error messages from tools}

For main threads:
Please use the tools. If no further actions required, please use the hang tool. *if no tool usage last round*
At {time}, you receive {new alerts}. These alerts are still ongoing. {ongoing alerts}.
These alerts are still not linking to an anomaly. {orphan alerts}
These anomalies are still not causally related. If necessary, you should check the topology and communicate via NAM. {anomalies with no upstream causes}.

For assistance threads:
Please use the tools. If no further actions required, please return the message via NAM. *if no tool usage last round*

Experience 7. Messages that highlight orphan alerts and anomalies without upstream causes improve the TC metric by maintaining all related events in agent memory. However, an excessive number of these notes can compel the Xilets to infer causality prematurely, a tendency that must be carefully managed.

While normally XiHe does not generate a text for its NCR causal graph, we have the following prompt for the evaluation. The same template works for CA+ to summarize the alert causal graph. This prompt template focuses the LLM only on selective root nodes based on the causal graph.

Prompt for evaluation

You are encountering an incident with the following causal relations.

The anomaly of {anomaly type} in {fault domain} causes {number of downstream anomalies}. The severity of this anomaly is {highest heuristic severity of the attached alerts}.
...

Summarize the root cause with anomaly type and fault domain of the incident. If one is not enough indeed, you may give at most two possible root causes. You do not need to show your thinking process.

B.2 Pre-Release Testing

Although a secured LLM serving platform exists within our production boundary, direct access was restricted during the design phase. Consequently, we locally deployed the open-source Qwen model via vLLM [42] in a pre-release test environment and test the new prompt and OKB on previous incidents retained.

Using the pre-release testing, we are able to look into the cases that the prompt or OKB optimization does not anticipate. For example, in our pre-release testing, we used to prompt the agent with "Your current anomalies without any causes have the causal schema like this. And these are the anomalies from your neighboring device, which has the following alerts. You may inspect their alerts, and build causal relations between the anomalies." Unexpectedly, the model misinterpreted the prompt, and created new anomalies for the alerts at other devices at a high rate (almost every time in one run). Despite the fact that these actions are unharmful (as NCR would reject them), it introduces extra token cost for generating the proposal to NCR. We quickly fixed the prompt from inspecting the *alerts* to *anomalies*. Learning from this experience, we also adopt a safety mechanism in NCR, where proposals of new anomalies can only be made by the Xilet of the anomaly's fault domain or its initial alert's signature.

The local setup remains fully compatible with production requirements, as the core capabilities utilized by XiHe, specifically prompt caching and tool-call parsing, are standard features supported by mainstream serving engines like vLLM, SGLang [84] and Mooncake [61].

C INTERACTION WITH OPERATORS

C.1 Overall Iterative Process

We exhibit the overall process of how we involve operators in our design process in Figure 17, which involves both questionnaire and consultative feedback from operators. The improvement of XiHe as shown in Figure 7(b) is achieved with OKB and prompting improvement.

C.2 Correlations of Quality Metrics

We use the Spearman correlation [63] between the sum of the quality metrics and the satisfaction level (measured as the number of positive questionnaire responses) to evaluate the relationship between quality metrics and interpretability. The Spearman correlation of these two metrics is 0.78. As implied by a USGS report [24], this demonstrates a semi-strong correlation between the two metrics. Considering the weight of accuracy to affect the operators' judgment, we believe that this shows a highly strong correlation between the quality metrics and the level of interpretability to operators.

C.3 Questionnaire

In our daily deployment, we ask the operators to fill out an optional questionnaire to help us improve XiHe. For the convenience of our operator, we keep the questionnaire as short as possible. Most of the questions accept quick yes-or-no answers. We also provide them with options for supplementary text feedback.

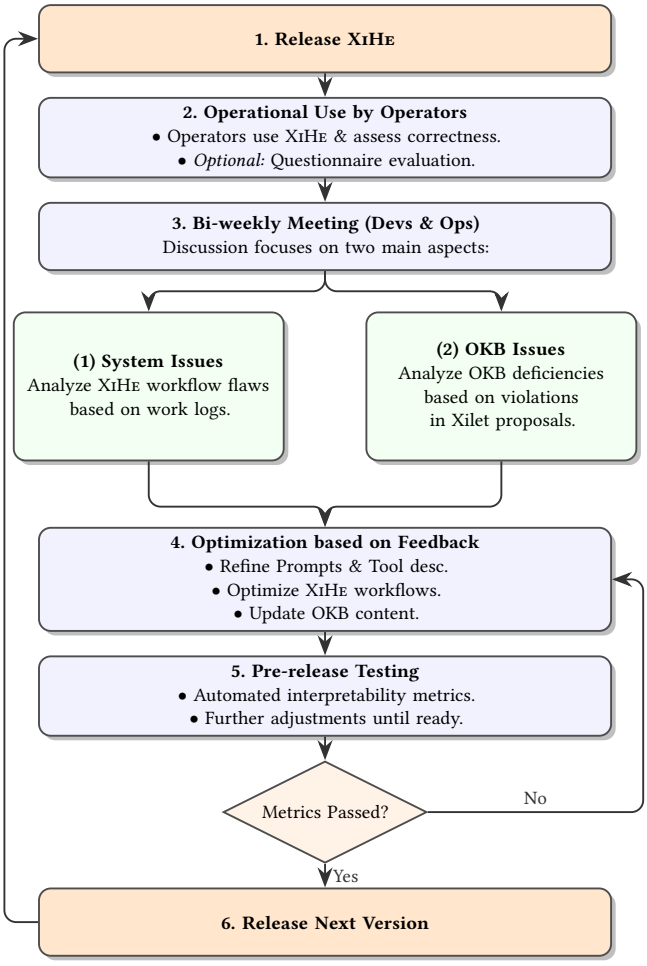


Figure 17: The iterative design process with operators.

Correctness:

❶ Is one of the root anomaly nodes given in the dashboard pointing to the root cause you care about?

❷ Are other root anomaly nodes indicators of noises or other possible concurrent failures?

❸ If your answer to question 1 is no, is the root cause reported as an alert, no matter triggered by telemetry or event?

Clarity

❹ Is the propagation chain from the root cause to the SLA degradation alert obvious?

❺ What portion of the other root anomaly nodes given in the dashboard are the propagated symptoms of the root cause anomaly?

Completeness:

❻ Are the key alerts of the incident included in the causal graph?

❼ Is the causal graph sufficient to conclude that the root node is the true root cause or not?

Soundness:

❽ Are there hallucinated anomalies or alerts?

❾ Are there unreasonable causal relations or symptoms of anomalies?

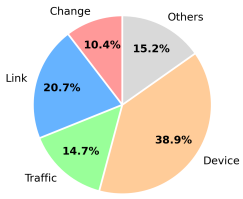


Figure 18: The portion of incident types of 624 incidents.

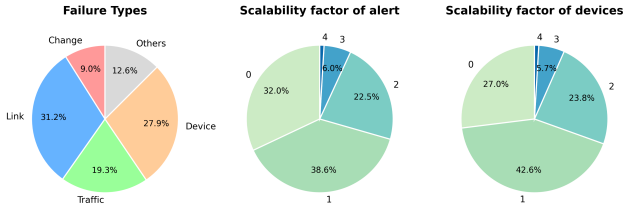


Figure 19: The portion of incident types and scales of 1067 incidents after SoP filtering. Incidents with larger amount of alerts and devices involved often come with higher priority and larger impact at service level. More than 130 incidents are reported with 100+ alerts (after filtering) or 100+ devices.

We receive questionnaire feedback from roughly 15% of incidents. We consider an operator satisfied with XiHe's output for an incident if the operator answers positively (for example, yes for ❶, no for ❸, less than 5 for ❺) to no less than seven of the nine questions above.

C.4 A/B Tests

We evaluate the time interval from the starting of the incident to the moment the operators take mitigation actions as the root-causing time. Our operation team consists of 20+ operators with different levels of expertise. In each group of on-call operators, we select two operators with the same level of expertise. One of them is instructed to use the older version of the operator dashboard, while the other is told to use the new version[4] respectively to ensure fairness. Generally speaking, XiHe achieves a warm welcome from operators. As shown in Figure 8, XiHe reduces the root-causing time by 12.2% for known issues and 27.6% for unknown incidents.

D EXTENSIVE EVALUATION DETAILS

D.1 Incident Details

In §9.1, the filtered 624 incidents exclude the incidents that require alerts from certain data sources, such as optical fiber information and full change log information. We exhibit the portion of each type of incident in Figure 18. In §9.3, the 1067 incidents are a superset of the 624 incidents, while also including 4 large-scale incidents taken place at other time. We exhibit the portion of each type of incident, and the scale of incidents in Figure 19. As shown in Figure 11, a non-negligible portion of incidents are of large scale and of high risk, where baselines do not perform well.

At the time of incident retention, the filtering mechanism for the "known issues" as introduced in §8.1 has not been adopted. As a

[4]The older dashboard has integrated other tools such as SkyNet and BiAn, while only the newer dashboard is equipped with XiHe.

result, these incidents do not distinguish the known and unknown ones.

D.2 LLM Serving Platform

In Table 2, we exhibit the price for the Qwen3-MAX model on our serving platform. For the input price, we have the original price with no prompt caching in the first place. Baselines like BIAN could not benefit much from the prompt caching, since its agents only perform one round of interaction. It also suffers from the higher level of per-token prices due to the long contexts for large-scale incidents. On the other hand, XIHE's requests always fall into the lowest price level, benefiting from the decomposed shorter contexts.

As we acknowledge in §9.5, the fluctuation of LLM serving is negligible due to the relatively short prompt of XIHE. However, we do observe that the request priority can affect its latency, depending on other co-running tasks on the platform. To ensure a fair evaluation, we set the request priority of XIHE and other baselines in the evaluation at the same level as in XIHE's production environment. However, we implement a strict request-per-second (RPS) limit, so that we do not interfere with other tasks co-running on the platform.

D.3 Full Intrepretability Results

We exhibit the interpretability metrics of XIHE in Figure 20. We show the improvement of XIHE against CA+ on each metric in Figure 20(c). The absolute values of SC_2 and SC_3 are smaller than SC_4, since the concentration of LCD is more obvious when the k is larger. Notably, XIHE achieves high SC improvements on other incidents, showing superior performance on the incidents out of design scope.

D.4 Ablation Study on Interpretability

We progressively add MR, NAM and NCR to OTA to show each component's effect on interpretability. Notably, OTA does not display causal graphs, and we cannot measure it with interpretability metrics. We exhibit the interpretability metrics in Figure 21. NAM improves the TC of alerts by 6.1% against OTA+MR, since a group of Xilets make causal proposals in parallel and add more causal links into the causal graph. NCR greatly improves the concentration of the propagation chain as the SCs rises from the OTA+MR+NAM setting by 43.0% to 47.1%, with the abstraction of anomalies.

The accuracy ablation in §9.1 shows the three components addressing different failure sources. The multi-agent design with NAM contributes the largest single gain (36.2% error reduction) by removing the context-window bottleneck: a monolithic agent must truncate alerts during an alert storm, whereas the swarm partitions them while NAM preserves the cross-scope dependencies. The OKB (alert-to-anomaly mapping) is the grounding layer present in every XIHE configuration, restricting the agents' search space to plausible network faults so the multi-round causal graph is trustworthy enough to build on. NCR contributes the final 21.1% as a structural validator that rejects topologically impossible or hallucinated edges and exposes the result as a verifiable graph.

E ADDITIONAL RELATED WORK COMPARISONS

In Table 1, we evaluate each system's adaptability to new failure modes with different symptom presentation or physical topology structure at the same level of low maintenance effort. The same

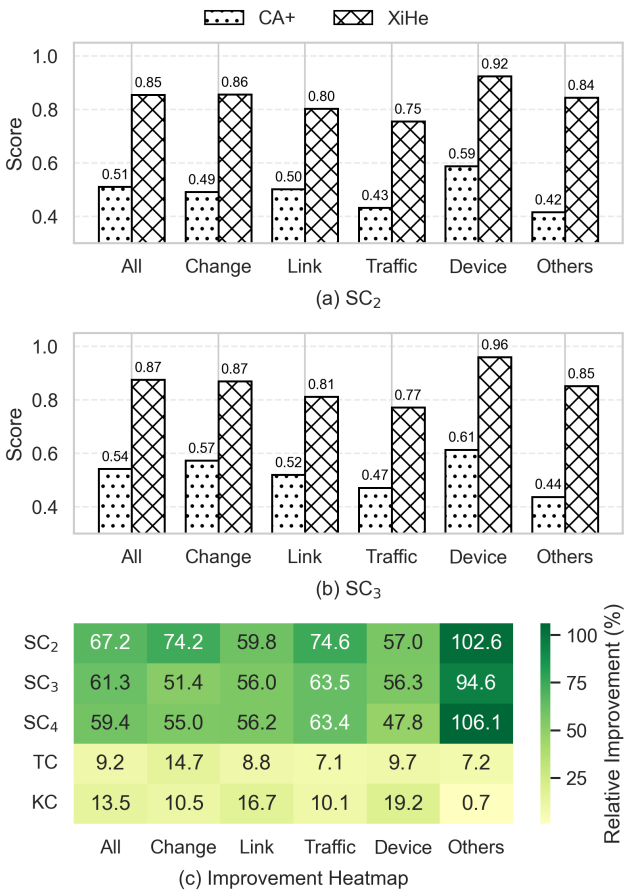


Figure 20: XIHE outperforms CA+ with better interpretability in all incident types.

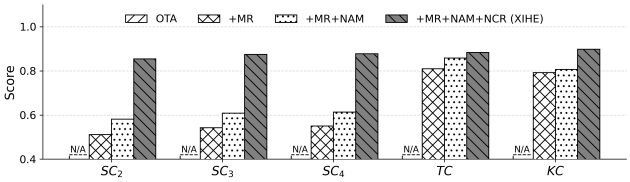


Figure 21: Ablation performance on interpretability.

works for the scalability. The intrepretability is output-oriented. We rate N/A for XPERT that gives no direct root cause result but a query command to search for relative incidents.

We further compare XIHE against the baselines in Table 3. The stability for any human-in-the-loop system is rated as N/A, as they incorporate an unfair benefit with more human effort. We take the pre-training (if applicable) and routine maintenance cost into consideration when evaluating the affordability. Notably, root-causing in physical networks normally presents higher maintenance costs than in microservice environments due to the complex and heterogeneous nature of the underlying infrastructure.

Despite the absence of human intervention and the difficulty of physical network root-causing, our proposed method, XIHE, outperforms existing baselines by achieving a superior balance across all

Table 2: Price Table for qwen3-max

Input tokens per request	Input price (per 1M tokens)	Output price (per 1M tokens)
0<Token≤32K	$1.2 (unhit: $1.5, hit: $0.12)	$6
32K<Token≤128K	$2.4 (unhit: $3.0, hit: $0.24)	$12
128K<Token≤252K	$3 (unhit: $3.75, hit: $0.30)	$15

Table 3: Extended comparison of related works.

Method	Env.	Stable	Affordable	Timely
Heuristics and Traditional ML				
SkyNet [74]	Phy.	✓	✓	✓
MicroCERCL [86]	Micro.	✗	✓	✓
Tool-Augmented Agent				
RCACopilot [9]	Micro.	✓	✓	✓
Xpert [37]	Micro.	N/A	✓	✓
OpenRCA [71]	Micro.	N/A	✗	✗
TAMO [83]	Micro.	✗	✓	✓
Multi-Agent Workflow				
mABC [81]	Micro.	✓	✗	✗
Cloud Atlas [70]	Micro.	N/A	✓	✓
BiAn [65]	Phy.	✓	✓	✓
OpsAgent [49]	Micro.	N/A	✗	✗
XiHe (ours)	Phy.	✓	✓	✓

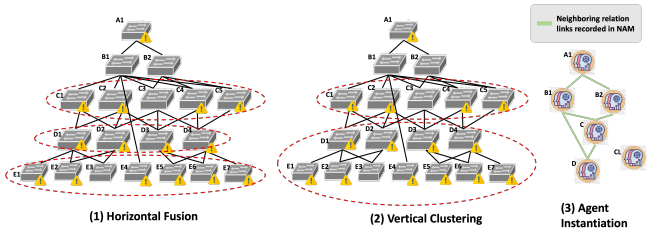


Figure 22: XiProxy instantiates the Xilets and dispatch alerts to them. CL means the signature-bound Xilet for change logs.

evaluation metrics. Notably, it excels in scalability and interpretability, proving its robustness in handling the intricate dynamics of physical network infrastructures where other methods often fall short. While XiHe shows superior performance to heuristics such as HotDevice, SkyNet and 007, we acknowledge that XiHe does not aim to replace them. These heuristic approaches remain valuable and highly efficient tools to aggregate the alerts to incidents and identify the boundaries of the Xilet swarms.

F RUNNING EXAMPLE OF XIHE

F.1 XiHe Processing

We provide a step-by-step example to show how XiHe handles the incident given in Figure 1. Note that the incident is simplified and only for demonstration purposes. We only consider a single parallel swarm in this section.

At the first step, XiProxy looks into the alerts, and aggregates the devices to form device-bound Xilets according to the memory limit of agents, as illustrated in Figure 22. A signature-bound Xilet for change logs is also created. Each Xilet then receives the corresponding alerts within its domain from the XiProxy.

Then, the swarm of Xilets raises their hypotheses and verifies them like in Figure 23. The knowledge (*i.e.*, alert-to-anomaly candidate mapping and anomaly causal schema) is automatically injected to each Xilet's memory, and is not shown in the figure. When Xilets find the information in their memory insufficient, they turn to NAM for other Xilets' insight. Specifically, the signature-bound Xilet uses predefined rules to filter out the change logs that are less relevant (for example, taking place in other parts of the network), and preserve its insight in its local memory. We do not show the device-bound Xilet for B2 in the figure, which have similar workflow as B1. Upon receiving the REQ from other Xilets, the Xilet for B1 and aggregated devices C starts their corresponding assistance thread as we described in Appendix A.2. In the real execution, **there is no synchronized interaction round** that we show in the figure.

Next, as shown in Figure 24, NCR post-processes the causal graph in NCR by removing the redundant edges, recovers the aggregated alerts and criticizes it with the quality metrics. Finally, XiHe sends a finalized formatted JSON file to the dashboard. The operators can then validate the root cause. Based on the impact on the network, they quickly exclude the possible root cause of the port shake on A1, and consider the software update failure on B1 as the root cause. In the evaluation setup, we further summarize the post-processed causal graph of NCR with LLM as discussed in Appendix B.1.

F.2 Frontend Exhibition

The dashboard for operators is shown in Figure 25a. It is integrated into our original dashboard. The incident description area details the panorama alerts and the results of other tools (*e.g.*, BiAn). The detail of the triggering alert also exhibits in the incident description area for the known incidents as described in §8.1. The causal graph area exhibits the top four layers of the original NCR causal graph. For further investigation and verification, the operators can click the figure and enter the causal graph's pop-up window.

Figure 25b shows an example of rendered NCR causal graph. The red blocks indicate the root anomaly nodes, while the yellow blocks are the derivative anomalies. The purple blocks are the raw alerts, attached to the anomalies. For trainless transition, we also annotate each directed link with the labels, such as *manifested as* (from anomaly to alert) or *propagated to* (from anomaly to anomaly).

The dashboard also renders another graph, which is a topology-only NCR, as shown in Figure 25c. Unlike original NCR, topology-only NCR combines all anomalies of the same fault domain into one anomaly node, and counts the number of anomalies in each fault domain. The fault domain with more than five anomalies will be marked with a little flame (can be seen in Figure 25a). The topology-only graph is helpful to the operators when the root-causing result is not accurate (*e.g.*, another port or NIC of the identified device). They can instead inspect the network entities around the hotspot fault domain and accelerate their speed to the true root cause.

F.3 Failure Mechanism of Case 5

In the unobserved traffic surge of Case 5 (§8.2), the absent *traffic* root alert forced each Xilet to fall back to the next-most-likely OKB

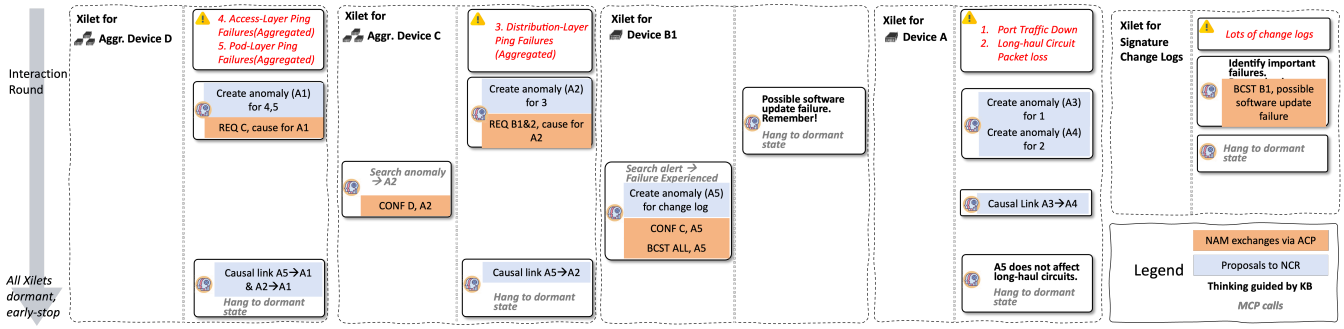


Figure 23: The illustration of execution of the a parallel swarm. The main thread (see Appendix A.2) is the rightmost thread of each Xilet. In the real execution, the starting time for each round of different threads are different.

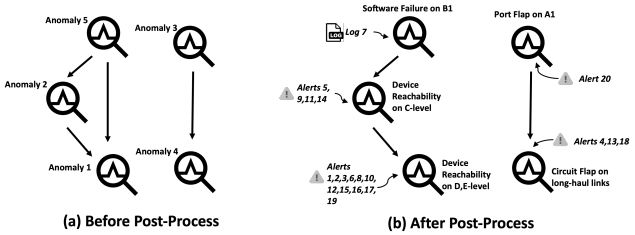
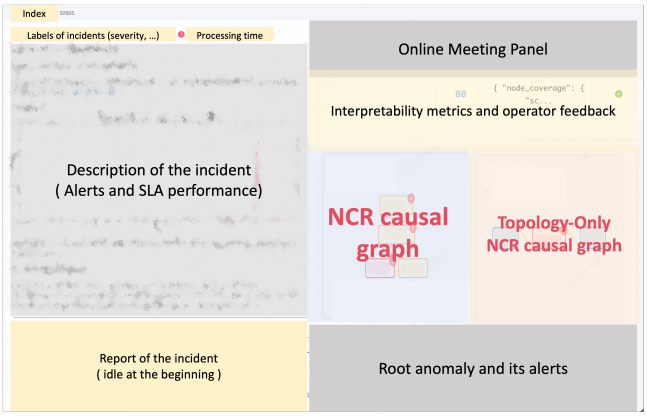


Figure 24: The illustration of NCR post-process and presentation. The post-process removes the causal link from anomaly 5 to 1, due to the existing path of anomalies 5⟶2⟶1. The causal graph after processing is shown to the operators.

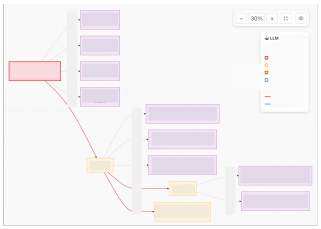
candidates for the *Packet Loss* signature (*e.g.*, *Link Degradation*), as the true cause emitted no anomaly to anchor on. The swarm thus composed a locally consistent chain rooted at a link anomaly; every constructed edge respected an OKB causal rule, so the NCR passed validation despite being wrong-rooted. The tell was structural rather than logical: the resulting graph showed low structural concentration, with correlated loss anomalies clustering in a single fault domain instead of converging on one dominant root, which is the signature operators read as a load problem rather than a physical fault.

G THE NAMING OF XIHE

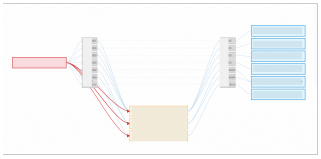
In Chinese mythology, Xi He (or Hsi Ho) is a solar deity and the mother of ten suns. She is responsible for illuminating the earth through the assistance of her children during the day. We named our system after Xi He to reflect its capability to illuminate the root-causing for complex incidents of diverse failure modes and reveal the propagation chain from root causes to SLA degradation symptoms. Drawing a parallel to Xi He's sun sons, XiHe utilizes Xilets to collaboratively discover the root cause of each incident.



(a) The operator dashboard for an incident integrated with the XiHe's final conclusion, NCR causal graph and operator feedback panel.



(b) Example of an NCR causal graph on its separate window.



(c) Example of a topology-only NCR causal graph detailed in Appendix F.2.

Figure 25: The operator dashboard with XiHe. The figures blur some information due to commercial privacy concerns.