INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY EVOLUTIONARY RESEARCH

Graph-Based Modeling of Service Dependencies for Predicting Failure Propagation in Distributed Systems

Nilesh Mutyam
Senior Software Developer, Walmart Global Tech, Dallas, TX, United States

* Corresponding Author: **Nilesh Mutyam**

Article Info

P-ISSN: 3051-3502
E-ISSN: 3051-3510
Volume: 05
Issue: 01
January - June 2024
Received: 13-12-2023
Accepted: 15-01-2024
Published: 17-02-2024
Page No: 113-116

Abstract
Distributed systems increasingly rely on microservice architectures whose runtime behavior is shaped by rapidly changing service-to-service dependencies. When a fault occurs, symptoms often spread non-locally across the dependency structure, obscuring the initiating component and delaying remediation. This paper presents a graph-based modeling approach that (i) constructs a dynamic service dependency graph from request traces and operational telemetry, (ii) learns a propagation-aware representation of services and interactions, and (iii) predicts the likelihood and extent of failure propagation under partial and noisy observations. The proposed design combines structural causality encoded in traces with temporal signals from service health indicators to produce actionable outputs: predicted blast radius, ranked impacted services, and attributed propagation paths. We further define evaluation protocols for propagation prediction, discuss deployment constraints (instrumentation, concept drift, and multi-tenancy), and outline a reproducible experimentation methodology aligned with modern AIOps practices.

DOI: https://doi.org/10.54660/IJMER.2024.5.1.113-116

Keywords: Microservices, distributed systems, service dependency graphs, failure propagation, graph neural networks, distributed tracing, AIOps, root cause analysis

1. Introduction

Diagnosing performance issues in microservices remains difficult because the same user-facing symptom can emerge from multiple interacting components and data modalities. [1]

Recent causal discovery methods show that modeling faults as interventions can improve root-cause reasoning in microservice telemetry streams. [2]

Operational governance is also shifting toward decision intelligence workflows that convert observability signals into repeatable, auditable actions rather than ad-hoc debugging. [3]

Trace-driven graph deep learning highlights that faults can be represented as propagation patterns over structured call graphs instead of isolated metric spikes. [4] Automation ROI studies further motivate proactive failure propagation prediction because earlier containment reduces incident duration and operational cost. [5]

2. Background and Motivation

2.1 Failure propagation as a graph problem

In microservice architectures, dependencies are not merely static call graphs; they are dynamic, workload-conditioned relations that change with deployments, routing, caching, retries, and partial failures. A fault in one node can propagate via request amplification, backpressure, shared resources, and cascading timeouts, producing symptoms in upstream services that appear more broken than the originating component. The core challenge is to infer where failures will spread next and which services will be impacted given incomplete early signals.

2.2. Trace-centric dependency modeling

Trace-based approaches show that request traces expose ordering and parent-child relations among service spans, providing a natural substrate for dependency graph construction. [6]

General graph anomaly detection literature provides a useful lens for handling dynamic graphs, heterogeneous attributes, and rare-event regimes typical of production incidents. [7]

Distributed tracing has also been used for automated fault localization by following error signals backward through service interactions. [8]

Implicit dependency learning with graph convolution demonstrates that learned service graphs can reveal abnormal service behavior not obvious from raw time series alone. [9]

2.3. Why prediction (not only localization) matters

Traditional root cause analysis focuses on localization after the incident is visible. Propagation prediction targets early warning and blast-radius estimation to trigger mitigations such as traffic shedding, circuit breaking, or selective rollback. Defect and fault prediction research in software engineering motivates learning from historical failure patterns to anticipate future incidents. [10]

3. Problem Formulation

3.1. System model

Let G_t = (V_t, E_t) denote a directed dependency graph at time window t, where V_t are services (or service instances) observed in the window and E_t are directed interactions inferred from traces and/or communication metadata. Each node v in V_t has features x_v(t) derived from telemetry, such as latency percentiles, error ratios, saturation indicators, and trace-derived structural descriptors. Each edge e = (u -> v) has features x_uv(t) such as call frequency, fanout, retry rate, and conditional latency shift.

3.2. Propagation prediction objectives

Given early observations from a time window t, predict: (1) Blast radius B_t subseteq V_t, the set of services likely to become impacted in t + Δ; (2) impact probabilities p_v(t + Δ) for each service v; and (3) attributed propagation paths, a ranked set of paths or subgraphs explaining predicted spread. This differs from root-cause localization because the optimization focuses on the future impacted set rather than only the initiating node.

4. Dynamic Service Dependency Graph Construction

4.1. Trace-derived structure

We build E_t primarily from distributed tracing, using span parent-child relations and cross-service edges (client span -> server span). For each edge u -> v, we estimate edge weight components: (a) intensity w_freq_uv, normalized call volume in the window; (b) latency shift w_lat_uv, deviation from baseline critical-path latency contribution; and (c) error coupling w_err_uv, conditional probability of errors in v given anomalies in u.

4.2. Handling dynamic topology and drift

Graphs drift due to deployments, feature flags, autoscaling, and routing policies. OpenTracing-based anomaly work shows that cloud tracing signals can shift with operational changes, requiring adaptive baselines. [11]

We therefore maintain a rolling baseline graph G_bar over a longer horizon and a change score delta(G_t, G_bar) that marks structural drift (new edges, vanished edges, and weight shifts).

4.3. Multi-modal augmentation

While traces encode interaction structure, many incidents are multi-modal. Service behavior models that combine traces with profiling metrics can capture context propagation more precisely than single-modality approaches. [12]

Distributed tracing can also support online failure detection pipelines that update models as new call paths emerge. [13]

NLP-based analysis of trace/log textual artifacts provides another channel for identifying anomalous behaviors that do not surface strongly in metrics. [14]

5. Propagation-Aware Learning Model

5.1. Intuition

Propagation prediction requires structural sensitivity (understanding how failures flow along dependencies) and temporal sensitivity (accounting for ordering and escalation such as queue buildup before timeouts).

Fine-grained causal graph approaches show that weighted causal structures can support more precise localization of responsible signals. [15]

We extend this intuition: rather than only inferring causal parents of a symptom, we learn a propagation operator that forecasts downstream impact probabilities.

5.2. Model sketch

We define a propagation network that consumes G_t and predicts node impact probabilities for t + Δ. A practical formulation is a temporal message passing model:

1. Edge-conditioned message passing: m_v = sum_{u:(u->v)
2. in E_t} phi(h_u, x_uv(t))
3. Node update: h_v' = psi(h_v, m_v, x_v(t)) Impact head: p_v(t + Δ) = sigma(W h_v' + b) where h_v is a learned embedding and phi, psi are learnable functions.

5.3. Explainability through path attribution

To support SRE decision-making, the model emits top contributing incoming edges for each high-risk node and a minimal subgraph (a propagation witness) explaining predicted blast radius.

Security-aware operations benefit from interpretable dependency reasoning [16] because failures and attacks can share observable signatures such as latency spikes, saturation, and error cascades.

6. Operational Integration

6.1. Incident lifecycle integration

A robust design integrates with alerting and remediation: (1) early detection (trigger); (2) propagation prediction (blast radius and risk ranking); (3) mitigation suggestion (traffic shaping, rollback candidates, isolation); and (4) post-incident learning (label refinement and baseline updates).

Multimodal dependency-graph approaches show that richer representations can improve diagnosis accuracy when failures are imbalanced and sparse. [17]

6.2. Graph pruning for scale
At large scale, running inference over full graphs per incident can be expensive. Reinforcement learning-driven pruning of service dependency graphs provides a strategy to keep only diagnostically relevant regions while preserving accuracy. [18]

6.3. Security and critical infrastructure considerations
Predicting propagation is also a resilience problem in smart infrastructure and public utility contexts, where cascading failures can have broader societal impact. [19]

7. Experimental Protocol
7.1. Data requirements and labeling
Propagation prediction can be trained with incident postmortems mapping impacted services, SLO violation windows labeling impacted sets, or fault-injection experiments in staging.
Temporal causal discovery can leverage ordering in time series to improve causal directionality assumptions in microservice telemetry. [20]

7.2. Baselines and metrics
Suggested baselines include topology-only heuristics (reachability and PageRank-like spread), anomaly localization models (predict root cause only), and causal-graph RCA methods with downstream expansion.
Suggested metrics include blast radius F1 over B_t, top-k impacted service recall, time-to-accurate-blast-radius (how early the model becomes reliable), and attribution fidelity (agreement with validated propagation paths).
Multi-bottleneck learning results indicate that realistic microservice performance issues often involve multiple interacting culprits, which is aligned with propagation prediction rather than single-root assumptions. [21]

7.3. Proactive prediction vs reactive localization
Failure prediction paired with root cause identification [22] can be implemented as a joint model over dynamic graphs and multi-modal attributes, enabling earlier intervention than post-hoc RCA.
Modal-independent learning strategies can reduce dependence on any single dominant telemetry modality, improving robustness under partial observability. [23]

8. Discussion
8.1. Why this is economically actionable
Propagation prediction can reduce toil by converting symptom storms into a prioritized impacted-service list and an explainable propagation witness.
Healthcare AI experience highlights that trustworthy, explainable automation is essential when decisions influence reliability-critical workflows. [24]

8.2. Temporal knowledge graphs as a complementary direction
Temporal knowledge graph fusion approaches suggest a natural extension: combine trace structure, deployment metadata, and evolving telemetry semantics into a unified temporal representation. [25]

8.3. Instrumentation constraints
Full instrumentation is not always feasible. Trace reconstruction without application modification [26] can reduce deployment friction while still supporting downstream graph modeling.

8.4. Broader safety context
Failure propagation prediction intersects with online safety concerns when cascading outages affect user trust, security posture, and platform reliability. [27]

9. Threats to Validity
Concept drift: dependency graphs and workloads change rapidly; models must adapt without forgetting.
Label noise: impacted service may be underreported when telemetry is incomplete.
Hidden dependencies: shared infrastructure (databases, caches, networks) can create propagation paths not explicit in traces.
Evaluation realism: staging fault injection may not reproduce production concurrency, retries, and tail behavior.

10. Conclusion
This paper described a graph-based approach for modeling service dependencies and predicting failure propagation in distributed systems. The central idea is to treat traces as dynamic structural evidence, augment them with telemetry attributes, and learn propagation-aware representations that forecast blast radius and provide explainable propagation paths. The resulting outputs can support earlier, more targeted mitigation than reactive diagnosis alone, while remaining compatible with real-world constraints such as drift, partial observability, and heterogeneous telemetry. Fault prediction research also supports the broader thesis that learning from historical system behavior improves preparedness for future failures. [28]

References
1. Hou C, Jia T, Wu Y, Li Y, Han J. Diagnosing performance issues in microservices with heterogeneous data source. In: Proc. ISPA-BDCloud-SocialCom-SustainCom 2021. 2021. Available from: https://www.cloud-conf.net/ispa2021/proc/pdfs/ISPA-BDCloud-SocialCom-SustainCom2021-3mkuIWCJVSdKJpBYM7KEKW/264600a493/264600a493.pdf.
2. Ikram A, Chakraborty S, Mitra S, Saini S, Bagchi S, Kocaoglu M. Root cause analysis of failures in microservices through causal discovery. In: Advances in Neural Information Processing Systems (NeurIPS 2022). Vol. 35. 2022. Available from: https://proceedings.neurips.cc/paper_files/paper/2022/hash/c9fcd02e6445c7dfbad6986abee53d0d-Abstract-Conference.html.
3. Sivva SD, Thalakanti RR, Bandari SSG, Yettapu SDR. AI-driven decision intelligence for agile software lifecycle governance: an architecture-centered framework integrating machine learning defect prediction and automated testing. Int J Eng Technol Comput Sci Inf Technol. 2023 Dec 30;4(4):167-72. Available from: https://www.ijetcsit.org/index.php/ijetcsit/article/view/554.
4. Chen J, Liu F, Jiang J, Zhong G, Xu D, Tan Z, *et al*. TraceGra: a trace-based anomaly detection for microservice using graph deep learning. Comput

Commun. 2023;204:109-17. doi: 10.1016/j.comcom.2023.03.028.
5. Kacheru G, Bajjuru R, Arthan N. The ROI of software automation: measuring time and cost savings. Int J Commun Netw Inf Secur. 2023;15(4):774-85.
6. Li Z, *et al*. Practical root cause localization for microservice systems via trace analysis. In: Proc. IEEE/ACM IWQoS 2021. 2021. p. 1-10. doi: 10.1109/IWQOS52092.2021.9521340.
7. Lamichhane PB, Eberle W. Anomaly detection in graph structured data: a survey. 2024. arXiv:2405.06172. Available from: https://arxiv.org/html/2405.06172v1.
8. Rios J, Jha S, Shwartz L. Localizing and explaining faults in microservices using distributed tracing. In: Proc. IEEE 15th International Conference on Cloud Computing (CLOUD 2022). 2022. p. 489-99. doi: 10.1109/CLOUD55607.2022.00072.
9. Tang H, Guo Y, Yang J, Chen Y. Microservice anomaly diagnosis with graph convolution network based on implicit microservice dependency. 2023. doi: 10.1145/3594315.3594354.
10. Gunda SK. Comparative analysis of machine learning models for software defect prediction. In: 2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India. 2024. p. 1-6. doi: 10.1109/ICPECTS62210.2024.10780167.
11. Khanahmadi M, Shameli-Sendi A, Jabbarifar M, Fournier Q, Dagenais M. Detection of microservice-based software anomalies based on OpenTracing in cloud. Softw Pract Exp. 2023;53(8):1681-99. doi: 10.1002/spe.3208.
12. Panahandeh M, Hamou-Lhadj A, Hamdaqa M, Miller J. ServiceAnomaly: an anomaly detection approach in microservices using distributed traces and profiling metrics. J Syst Softw. 2024;209:111917. doi: 10.1016/j.jss.2023.111917.
13. Mazraemolla ZP, Rasoolzadegan A. An effective failure detection method for microservice-based systems using distributed tracing data. Eng Appl Artif Intell. 2024;133(Pt F):108558. doi: 10.1016/j.engappai.2024.108558.
14. Kohyarnejadfard I, Aloise D, Azhari SV, Dagenais MR. Anomaly detection in microservice environments using distributed tracing data analysis and NLP. J Cloud Comput. 2022;11(1):25. doi: 10.1186/s13677-022-00296-4.
15. Xin R, Chen P, Zhao Z. CausalRCA: causal inference based precise fine-grained root cause localization for microservice applications. 2022. arXiv:2209.02500. Available from: https://arxiv.org/abs/2209.02500.
16. Pittala SK, Ashok VKC. A new era in security: bridging information security and cybersecurity. Int J Multidiscip Futur Dev. 2023;4(1):69-72. doi: 10.54660/IJMFD.2023.4.1.69-72.
17. Zhang S, *et al*. Robust failure diagnosis of microservice system through multimodal data. IEEE Trans Serv Comput. 2023;16(6):3851-64. doi: 10.1109/TSC.2023.3290018.
18. Ding R, *et al*. TraceDiag: adaptive, interpretable, and efficient root cause analysis on large-scale microservice systems. In: Proc. ESEC/FSE 2023. 2023. doi: 10.1145/3611643.3613864.
19. Ashok VKC. Cybersecurity for smart infrastructure and public utilities. Int J Multidiscip Res Growth Eval. 2023;4(2):947-9. doi: 10.54660/IJMRGE.2023.4.2.947-949.
20. Lin CM, Chang C, Wang WY, Wang KD, Peng WC. Root cause analysis in microservice using neural Granger causal discovery. In: Proc. AAAI 2024. 2024. arXiv:2402.01140. Available from: https://arxiv.org/abs/2402.01140.
21. Somashekar G, Dutt A, Adak M, Lorido Botran T, Gandhi A. GAMMA: graph neural network-based multi-bottleneck localization for microservices applications. In: Proc. The ACM Web Conference (WWW 2024). 2024. doi: 10.1145/3589334.3645665.
22. Rouf R, *et al*. InstantOps: a joint approach to system failure prediction and root cause identification in microservices cloud-native applications. In: Proc. ICPE 2024. 2024. doi: 10.1145/3629526.3645047.
23. Tao L, *et al*. Giving every modality a voice in microservice failure diagnosis via multimodal adaptive optimization. In: Proc. ASE 2024. 2024. doi: 10.1145/3691620.3695489.
24. Kacheru G. Revolutionizing healthcare: the role of artificial intelligence in clinical practice. J Comput Anal Appl. 2023;31(4):1546-54. Available from: https://eudoxuspress.com/index.php/pub/article/view/3270.
25. Zhang S, *et al*. No more data silos: unified microservice failure diagnosis with temporal knowledge graph. IEEE Trans Serv Comput. 2024;17:4013-26. doi: 10.1109/TSC.2024.3489444.
26. Ashok S, *et al*. TraceWeaver: distributed request tracing for microservices without application modification. In: Proc. ACM SIGCOMM 2024. 2024. doi: 10.1145/3651890.3672254.
27. Pittala SK. Cybersecurity and online safety: a critical asset in the information era. J Front Multidiscip Res. 2023;4(1):576-9. doi: 10.54660/JFMR.2023.4.1.576-579.
28. Gunda SK. Fault prediction unveiled: analyzing the effectiveness of random forest, logistic regression, and KNeighbors. In: 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India. 2024. p. 107-13. doi: 10.1109/ICSSAS64001.2024.10760620