Journal of Research in Science and Engineering (JRSE)
ISSN: 1656-1996 Volume-8, Issue-5, May 2026
DOI: 10.53469/jrse.2026.08(05).06

Enhancing Service Reliability via Graph Reinforcement Learning: Real-Time Dependency Mapping and Failure Prediction

Rameshbabu Lakshmanasamy

IEEE Member, USA

Abstract: ***In large-scale distributed systems with numerous workflows and microservices, traditional service dependency mapping approaches rely on static graphs that fail to capture real-time changes, leading to delayed incident detection and prolonged downtime. This research explores Graph Reinforcement Learning (GRL) as a dynamic solution for modeling inter-service dependencies and predicting failure propagation in real time. By leveraging real-time telemetry data and historical incidents, GRL continuously updates dependency graphs, reducing Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR). The paper further discusses implementation challenges, including computational complexity and scalability, and proposes solutions such as hierarchical clustering and distributed processing. The findings suggest that GRL significantly enhances system resilience, making it a valuable tool for modern reliability engineering.***

Keyword: Graph Reinforcement Learning, service reliability, failure prediction, site reliability engineering, dynamic dependency mapping

1. Introduction

Today's cloud, SaaS, and other large-scale applications result from distributed systems controlling the attention of various fields. These systems are defined as thousands of workflows and tens of thousands of interconnected microservices that are constantly evolving in most cases. Therefore, tracking service dependency mappings and identifying root causes of failures is challenging.

The traditional approach to SRE involves using dependency graphs or hand maps, which are outdated in assessing the dynamic nature of the services and their interactions. Engineers face the major problem of identifying which dependency is responsible for failure in such circumstances; hence, the time taken to rectify the situation is enormous. This leads to higher values of Mean Time to Detect, i.e., the time taken before a failure is even detected by the customer, and Mean Time to Recover, which hampers system reliability and customer satisfaction. Graph Reinforcement Learning (GRL) is a promising strategy for specifying and predicting the service dependencies and failures in real time. Hence, using the real-time service logs and previous data of the incidents helps update the dependency graphs and defines which microservices or part of the infrastructure will likely fail next in GRL. Compared with conventional approaches which are usually static, GRL has adaptive, automated, and predictive features that aid organisations in preventing failures from going out-of-hand.

This research is significant as it addresses the limitations of static dependency mapping in large-scale distributed systems. By leveraging GRL, it proposes a proactive approach to failure prediction, potentially transforming incident management in cloud computing, SaaS, and large IT infrastructures. The study critically examines existing methodologies through service dependency mapping, proposes an innovative GRL-based framework for real-time failure prediction and dynamic dependency graph updates in IT systems, and evaluates MTTD and MTTR outcomes, demonstrating GRL's efficacy in reducing downtime and facilitating in-depth root cause analysis.

Some issues associated with implementing soft voting are computation overhead and scalability.

2. The Problem: Static Service Dependency Graphs and Their Limitations

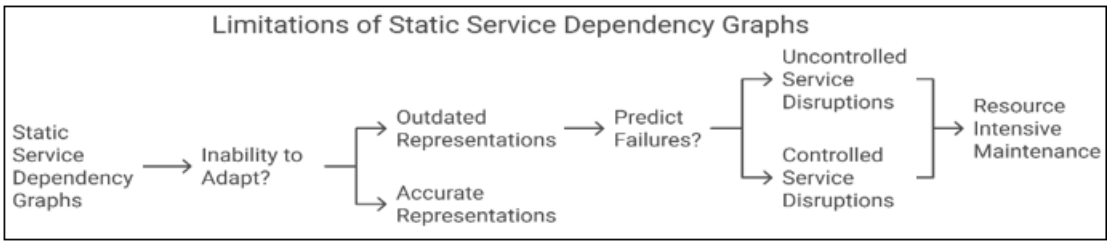
Now, Site Reliability Engineering (SRE) practices in modern businesses heavily depend upon service dependency graphs to understand the relationships between different service microservices and the infrastructure components. These graphs are the fundamental monitoring system for health monitoring, failure diagnosis, and performance optimization instruments. The dependency graph of service interactions that SRE teams traditionally use is based on manually created or statically generated graphs (Narapureddy, n.d.). However, traditional approaches tend to fall short as systems fall under thousands of workflows and tens of thousands of microservices.

2.1 Challenges with Static Dependency Graphs in Large-Scale Distributed Systems

One of the main challenges is that they cannot cope with changes to dynamic services. Due to the high fluid nature of modern microservices architecture, which consists of frequent updates and deployments, and deprecations, there is a strong need to avoid any architectural side affects you might have incurred efficiently. However, as the service mappings become static, they quickly become outdated and thus provide an inaccurate representation of the service relationships. These graphs can be written without continuous updates, and they fail to accurately diagnose failures when they do not reflect the current system state.

A further hurdle that is critical in limitation is the impossibility of predicting failure in real time. Static graphs refer to historical points rather than being predictive.

Therefore, they do not include telemetry data or past failure patterns and do not predict future problems until too late. Therefore, service disruptions continue to propagate uncontrollably, causing widespread outages and extended downtime without the ability to predict them accurately. Keeping static dependency mappings is also expensive and consumes many resources. Constant updating of the dependency graphs becomes necessary while operating large-scale environments with thousands of microservices. As a result, there are more operational costs and time-consuming troubleshooting processes. Dependency graphs need to be maintained accurately, and that prevents them from allocating engineering resources to more strategic initiatives.



2.1.1 Inability to Adapt to Dynamic Service Changes

Static service dependency graphs cannot reflect evolving service architectures in real time and are hence inadequate for manipulating service architectures. Microservices are updated, deployed, and retired often and quickly in large-scale environments, leaving dependencies changing rapidly (Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R., 2021). Dependency mappings done traditionally are never automatically updated, which makes them obsolete over time, making failure diagnostics inaccurate and incident resolution ineffective.

Services interact dynamically, and workloads disbursement, scaling pattern, and infrastructure changes (Fan et al., 2022) will shift the interdependency. These fluid transitions make static graphs inadequate, and SREs rarely have the correct information at hand if an incident arises. Since teams cannot pinpoint the origin of failures without an accurate and up-to-date representation of service relationships, they end up experiencing delays in mitigation and long downtimes.

A second fundamental issue is the complexity that the new DevOps practices require, in which rapid deployments and continuous integration (Tam, Ros, Song, Kang, & Kim, 2024) should occur in as little time as possible. Due to its high deployment frequency, the service landscape constantly changes, so the static graph cannot depict it. As such, manual tracking or ad hoc updates have become essential for engineering, increasing operational overhead and injecting additional human error.

The fact that static graphs cannot adapt to dynamic service changes also affects predictive maintenance efforts. As they lack live monitoring data and historical incident patterns, these graphs enable only the retrospective view but not forward-looking insights (Farahmand, Xu, & Mostafavi, 2023). This reactive approach fails only after such failure impacts system performance rather than managing risk that could develop into failures.

However, as with these properties, it is obvious that an alternative approach that dynamically updates the service dependencies in real time and integrates predictive analytics is needed to avoid failures before they happen.

2.1.2 Lack of Real-Time Failure Prediction

Static service dependency graphs primarily function as historical snapshots rather than predictive tools. They offer a retrospective view of system architecture but fail to integrate real-time telemetry data, anomaly detection, or historical failure patterns to anticipate future disruptions (Fan, Zhang, & Yu, 2022). As a result, incident response teams are forced into a reactive approach, addressing failures only after they have already impacted system performance (Tam et al., 2024).

Site reliability engineers (SREs) cannot foresee cascading failures across microservices without real-time failure prediction. A single point of failure in a critical service can propagate through dependencies, causing widespread outages, yet static graphs remain incapable of forecasting such scenarios (Farahmand, Xu, & Mostafavi, 2023). This limitation extends resolution time, requiring engineers to manually trace failures rather than leveraging predictive insights.

Moreover, modern distributed systems generate vast amounts of observability data, including logs, metrics, and traces, but static graphs fail to leverage this information for real-time analysis (Li, Liu, Zhang, & Fu, 2024). Without integrating live system behaviour, these graphs cannot dynamically adjust to evolving service states or identify early warning signs of degradation. Consequently, organizations experience increased mean time to detect (MTTD) failures, as alerts are often triggered too late for preventive action.

Given the increasing complexity of cloud-native architectures and the need for proactive reliability strategies, relying solely on static graphs presents a significant operational risk. A transition toward dynamic, learning-based models that continuously update dependency mappings and predict failure propagation is essential to enhancing service resilience and minimizing downtime.

Section	Details
Problem	Static Service Dependency Graphs and Their Limitations
Importance of Dependency Graphs	• Used in SRE practices to understand relationships between microservices and infrastructure. • Essential for health monitoring, failure diagnosis, and performance optimization.
Challenges with Static Dependency Graphs	• Cannot handle dynamic changes in services. • Become outdated quickly due to frequent updates and deployments. • Fail to predict failures in real-time. • Maintenance is costly and resource-intensive.
Inability to Adapt to Dynamic Changes	• Static graphs do not update automatically with evolving service architectures. • Frequent updates in microservices make static mappings obsolete. • Manual updates lead to delays and higher operational overhead. • Lack of real-time data affects failure diagnostics and incident resolution.
Impact on Predictive Maintenance	• Lack of live monitoring data and historical patterns limits proactive risk management. • Reactive approach leads to addressing failures post-impact.
Lack of Real-Time Failure Prediction	• Static graphs function as historical snapshots without real-time telemetry or anomaly detection. • Cannot foresee cascading failures across microservices. • Prolongs incident resolution times due to reactive approach.
Need for Alternative Approaches	• Requires dynamically updating service dependencies in real-time. • Integration of predictive analytics to prevent failures proactively.

2.1.3 High Operational Costs in Maintaining Manual Mappings

Continuous manual effort is needed to maintain accurate service dependency maps in large-scale distributed systems, and it is very resource-intensive and inefficient. Organizations must dedicate their engineering teams to tracking changes, updating documentation, and maintaining service dependencies correctly across time (Farahmand et al., 2023). As microservices continue to proliferate in application architectures, evolve, and interoperate with each other in a dynamic and distributed fashion among their architecture's many environments, which include, but are not limited to, cloud-native platforms, hybrid deployments and even third party APIs (Zhang & Cheng, 2020), this becomes increasingly unsustainable.

As systems scale, the burden of operationalizing these mappings scales exponentially. An update to a new microservice, API, or stack update requires changes to the existing dependency graphs. Despite being frequent as deployments in the DevOps world become very frequent, such updates (Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R., 2021) always lag behind the actual system changes, so they become outdated or wrong. This requires engineers to verify dependencies upon incidents manually, extend MTTR during an incident, and slow service recovery.

Besides direct engineering costs, static dependency graphs also bring in unseen inefficiencies in incident management. As these mappings are not automated processes, SRE teams need to go through multiple sources of information, including logs, dashboards, and legacy documentation, to diagnose service failures (Li, Liu, Zhang, and Fu, 2024). The fact that troubleshooting is broken makes it slow and leads to the situation where the root cause cannot be spotted, leading to prolonged downtime.

In addition, organizations that are relying on manual dependency mapping are prone to inconsistencies across teams. Knowledge silos and inconsistencies arise since dependencies are documented in different formats by different engineering groups (Tam et al., 2024). There is no central, dynamically updated mapping that makes collaboration amongst teams harder and makes it more challenging to resolve system failure faster.

2.2 Consequences of Static Dependency Graphs

The disadvantages of static dependency graphs are critical when it comes to managing the effect of each component on other components, which in turn affects the efficiency of incident control and reliability of the system.

The second-time consequence is long MTTD and MTTR: When an outage happens, the engineers use out-of-date mappings to locate the source of the problem, consequently prolonging the overall time required to detect and resolve the issue (Katragadda et al., 2021).

Lack of Real-time Failure Propagation Model: Work teams' inability to prioritize the services impacted by failures increases service downtime and multiple failures (Li et al., 2024). Higher Downtime Costs: Those industries where downtime is costly (such as finance and healthcare, cloud services), a small time lag in failure identification may result in huge revenue loss and tarnished reputation (Li, Jiao, & Yang, 2023).

In view of this, real-time issues require adaptive and intelligent approaches to modifying dependency graphs and propagating failures. Graph Reinforcement Learning (GRL) can mitigate these drawbacks as a self-updated, predictively learning solution for service dependency mapping (Hui et al., 2021).

3. Graph Reinforcement Learning: A New Approach to Service Dependency Mapping

Reinforcement learning (RL) is among the machine learning methodologies that allow the systems to decide on the best course of action with the help of experiences. With graph-based models incorporated, Graph Reinforcement Learning (GRL) can learn dynamic dependency of LS-DSs (Working' and Zhang bin, 2021; Fan, Zhang, & Yu, 2022). While traditional dependency graphs are usually designed to be updated gradually and frequently lose their relevance quickly, GRL reconstructs the service relationships dynamically by

using historical incident data and continually analyzing telemetry information triggered in actual operations (Tam et al., 2024).

Standard SRE processes do not work with rigid mappings and cannot keep up with the system's flexibility in a microservices architecture. On the other hand, GRL adapts the GRL model of the service dependency continuously through practice and experience gained from failure, propagation of failures and performance indices (Farahmand, Xu, & Mostafavi, 2023). This adaptive learning process enables GRL models to build a service dependency graph that is ever dynamic. The incident response teams continuously work with the most updated and correct system map (Zhang & Cheng, 2020).

Another capability that GRL has is the ability to use preceding events for failure prognostication and determine the fundamental cause. Unlike other models that handle a failure as an isolated occurrence, GRL learns recurring dependency patterns and shares this feedback with future models (Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R., 2021). Thus, knowing which services are most impacted by failures in certain circumstances, engineers can be notified of complex dependencies in advance of failures (Li et al., 2024).

However, another advantage is that it can also recognize indirect requirements, that is, the requirements derived from other requirements. Service interactions in distributed systems are complex, making failure patterns harder to detect with static graphs (Li et al., 2023). Unlike other failure models, GRL models can review the failure chains and identify some dependencies that can be uncovered but are not necessarily described in the documentation. This capability makes the response to the incident much more effective as it can detect paths where the failure may extend and therefore will remain unnoticed (Hui, Yan, Chen, & Ku, 2021).

It also adapts automatically to changes to the new number or infrastructure of services. New microservices and overall organizational changes occur, and GRL adapts by adding these to its dependency graph without further user input (Tam et al., 2024). Further, mapping management is automated to avoid relying on reliability engineers to constantly update and maintain its entries so that they can engage in other advanced-level optimization and reliability plans (Farahmand, Xu, & Mostafavi, 2023).

Thus, integrating GRL into service dependency mapping will add another capability to the current static graph models and help the organization make better predictions with the least value of MTTD and MTTR of the failed system. This is a big step towards the self-learning of reliability engineering for dependability systems, which provides dependency maps and detects when something is about to go wrong with the end-user.

4. Predicting Failure Propagation with GRL

Failure propagation in distributed systems can be viewed and analyzed as the problem in graphs: nodes are services interconnected with edges depending on their dependencies. If a specific microservice is not performing as expected, a negative impact will negatively affect other affiliated services. Conventional approaches cannot predict such disruptions since the employed graphs cannot continuously change in response to dynamics (Li et al., 2024).

It can be averted before mishaps transpire with the aid of graph reinforcement learning (GRL). GRL always uses historical incident data to determine which services are more exposed to failure occurrences (Tam et al., 2024). This way, the possibility of shifting the dependence weight is introduced, which depends on the current schedule of services and infrastructure configuration.

Organizations should consider training GRL models on historical incident data to improve their output. These models study historical service disruptions, recognize the failure domino effect, and continuously improve decision-making processes (Farahmand, Xu, & Mostafavi, 2023). Conversely, GRL can return application-level failure predictions by dynamically updating its service relationships.

Indeed, among the main advantages of GRL, using the FMEA as an example, it can be concluded that one of the key benefits is the analysis of potential failures that can eventually lead to large-scale blackouts. Thus, it plays a crucial role in helping SRE teams predict and prevent service-related issues, something made possible by the risk scores GRL assigns to the numerous services (Zhang & Cheng, 2020). This can predict, cut down on Mean Time to Detect (MTTD), hence Mean Time to Recovery (MTTR), since it eases faster and informed response in case of an incident.

Another way to support it will be to present a case or simulation example that illustrates the possibility of applying GRL for failure propagation prediction and suppression. Specifically, providing service engineers with a set of failures that occurred to the system under test and comparing the results GRL produced against traditional static dependency graphs would allow quantifying the degree of improvement in terms of detection and response value.

5. Real-Time Incident Detection and MTTR Reduction

GRL also improves the real-time identification of incidents using the learned service dependency and failure patterns. Today's monitoring systems are based on alerting and static dependency tables, which increases the time it takes to identify the actual failure and consequently the Mean Time to Detect (MTTD) (Li, Jiao, & Yang, 2023). While comparing GRL with other approaches, it is important to know that it dynamically updates service dependency graphs with the possibility of failure identification prior to incidents.

Instead of discussing asset configurations, dependency graphs show the system's state based on the real-time telemetry data in GRL. This capability helps more accurate failure localization, which in turn helps engineers spend less time investigating the causes of an occurrence (Fan et al., 2022). In the case of a security incident, GRL can identify how it spreads and might indicate services at high risk and possible ways to deal with the situation based on past events (Tam et al., 2024).

That is why one of GRL's significant benefits is the possibility of automating the Root Cause Analysis (RCA). To avoid the need for manual debugging of the system, GRL uses past failures to predict other possible failures that may occur in the system. This minimizes the MTTR as engineers know which micro-service, infrastructure part, or external service is probably causing the issue (Farahmand et al., 2023).

Specifically, for the practical implementation of GRL, it can be wrapped along with the present tools such as Prometheus, Datadog, or OpenTelemetry. These platforms gather many metrics, logs, and traces which can be used as training data for GRL models. Thus, incorporating GRL-driven analytics in these tools helps improve the observability stack of any organization and moves from a reactive mode to a more proactive mode of incident identifying and mitigation (Zhang & Cheng, 2020).

6. Implementation Considerations and Challenges

There are some challenges involved when in integrating Graph Reinforcement Learning (GRL) for service dependency mapping and failure prediction in real-time operation:

Data sources: As mentioned in the works by Li, Jiao, and Yang (2023), GRL uses data from logs, observability metrics, and traces. Training and inference need deep failure data of structures, history, services, or changes in dependency over time. Employing other observability tools such as Prometheus, OpenTelemetry, and Jaeger can further extend GRL's capability of identifying dynamic dependency between services, as Tam et al. (2024). This additional feature renders GRL models different from static dependency maps, and their computation involves continuous processing of streaming telemetry data and dynamics of their predictions, which considerably consume many computation resources (Fan, Zhang, & Yu, 2022). That is even more difficult in large-scale distributed systems with millions of these microservices, where tracking high-dimensional dependencies comes with a heavy processing cost (Farahmand, Xu, & Mostafavi, 2023).

Scalability issues: GRL must perform well in settings with tens of millions of dependencies, whereas conventional RL approaches are applied in training and inference. However, a failure prediction model based on genuine raw language might cause latency when it is not optimized, making it less effective in a real-life situation (Zhang & Cheng, 2020).

Solutions: It is possible to address these challenges in the following ways. Most GRL models can be distributed to accomplish computations over various nodes leading to enhanced efficiency. Real-time inference can be served through edge computing so that cloud resources would not be utilized much and the valuable services would take less time to respond (Li et al., 2024). More specifically, in terms of scalability, it is possible to enhance GNN improvements, for example, the hierarchical clustering, as the microservices related to each other can be grouped and the computational hardness may be decreased (Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R. 2021).

By implementing these challenges, the organizations can quickly deploy the GRL-based service observability as a near real-time failure prediction system that goes beyond quick alerts and transforms into a proactive self-learning ground for improving the overall Mean Time to Detect (MTTD) to Mean Time to Recovery (MTTR).

Aspect	Details
Challenges	• **Data Sources:** Requires deep failure data (structures, history, services, and changes in dependencies) and integration with observability tools like Prometheus, Open Telemetry, and Jaeger. Continuous processing of streaming telemetry data consumes significant computational resources, especially in large-scale distributed systems. • **Scalability Issues:** Handling tens of millions of dependencies efficiently. Raw language-based failure prediction can cause latency if not optimized. *Reference:* Zhang & Cheng (2020).
Solutions	• **Distributed Computation:** Utilizing multiple nodes for efficient processing. • **Edge Computing:** Reduces cloud resource usage and response times. • **Scalability Enhancements:** Hierarchical clustering of microservices to reduce computational complexity. *References:* Li et al. (2024); Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R. (2021).
Benefits	It enables GRL-based service observability for near real-time failure prediction and improves Mean Time to Detect (MTTD) and Mean Time to Recovery (MTTR) by transforming alerts into proactive, self-learning systems.

7. Future Directions and Research Opportunities

These approaches can be united in Graph Reinforcement Learning (GRL) which provides a perfect solution for dynamic service dependency mapping and failure prediction. Nevertheless, some issues should be addressed to improve the system's performance, implemented in a larger scale, and utilized in other domains:

Adding anomaly detection to GRL model: Unsupervised anomaly detection circuiting with GRL helps in early failure prediction of a service since it is detected before it grows to be an issue of the system (Li et al., 2024). With autoencoders, variational inference, and self-supervised learning, GRL models can successfully filter out much noise to find correlation between service interactions and thus minimize false positives and increase accuracy in its predictions (Tam et al., 2024).

Adoptions beyond IT infrastructure: The case of GRL has significant potential in managing IT service management; however, more research is possible in other domains such as supply chain, manufacturing, smart grid, etc. For instance, in logistics, GRL can capture the dependency structure of suppliers, determine potential disruptions and allocate resources efficiently (Fan et al., 2022). In the same way, in smart manufacturing, GRL could improve the analytical

framework of the intelligent maintenance and predictive maintenance model by forecasting the equipment failures based on the real-time data in the sensors (Farahmand et al., 2023).

Possible enhancements in MARL for distributed systems: Current distributed environments may contain tens of thousands or even hundreds of thousands of self-contained services that function concurrently. GRL can let the MARL independent agents cooperate and share derived dependencies within related system components, as described in Zhang & Cheng (2020). Again, studies in MARL to advance GRL could perhaps lead to improved collaboration between services, load balancing, and failure handling mechanisms (Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R., 2021).

Future works aim to make GRL more efficient regarding scalability, computational complexity, and timely adaptability to enhance service observability. Thus, with the development of the concept of AI-based observability, GRL can become a solid foundation for the resilience of infrastructure in IT systems, making them self-adapt and self-sustain in the community to minimize losses due to failures.

8. Discussion

Real-life service interdependency graphs and their evaluation with other approaches introduce the problem of less effective analyses. That is why developing more flexible approaches in current Site Reliability Engineering (SRE) is necessary. As will be discussed later, traditional dependency graphs that primarily have a historical focus and are more difficult for large software systems, and especially for microservices-based architectures. The paper shows that the system cannot predict changes to real-time conditions, failure cascading, and optimal incident resolution since these aspects can devastate the system since they have leaked through as major unsolved problems.

One of the conclusions drawn from the work presented is that static graphs are unsuitable in the defined context since they appeared designed for setting up a system where the services remain static as they are, constantly being updated, deployed and retired from active service. This contributes to the rise in MTTD and MTTR because engineers need to identify dependencies and the root of a failure on their own. Moreover, the probability of these mappings is high and they require a substantial number of engineers to maintain them while addressing more critical reliability aspects. These aspects are compounded by the absence of timely failure prognosis since the breakdowns worsen before they can be controlled.

The shift from static graph mapping to the Graph Reinforcement Learning (GRL) mapping is pioneered in the current study and shifts the paradigm of reliability engineering. Given that, GRL offers a way to update service dependencies depending on the context of execution, incorporate telemetry data, and predict the failure cascading effect, which makes it a good choice for addressing the problems of utilizing static graphs. As such, GRL models employ past accidents and real-time information to create more proactive approaches to system reliability issues where an organisation can identify possible failure before it happens on a large scale.

Furthermore, using GRL to automatically complete both RCA and incident detection also impacts the SRE work model. This contrasts with static mapping techniques, where the mappings are updated manually and checked for every error that may occur, while GRL dynamically adjusts the dependency graphs after learning about the system's behavior. It increases the visibility of the service interactions and hence quick identification and response to the incidents besides efficiently utilizing the available resources. Moreover, integrating GRL with other monitoring tools such as Prometheus, Datadog, and Open Telemetry enables additional observability and improves predictive analysis for handling incidents.

Therefore, the understanding of GRL-based service dependency mapping is still theoretical, and there are still some barriers to its implementation. The data generated in this case mean that a constant stream of telemetry data must be processed, and the high-dimensional dependencies involved in a large-scale system can become demanding when it comes to computation. Also, it is important to note that updating the set of historical incident patterns and its availability in training GRL models is crucial to ensure accuracy. Despite this, there are still obstacles associated with utilization of such data due to limited access to reliable machine data observability tools and efficient cloud environments. In summary, transitioning from the TDG-based approaches to GRL-based ones significantly improves SRE practices. Through picking real-time learning, failure prediction, and automated dependency management, it will be easier to make the system more resilient and have minimal downtimes, together with having the proper response to a given incident. More work should be done on improving GRL for scalability, and in using it in large-scale programs, and practicing using AI as a substitute for observation and integration with traditional/standard analytical methods for software management.

9. Conclusion

This study highlights the shortcomings of static dependency mapping in modern distributed systems and proposes Graph Reinforcement Learning (GRL) as a more dynamic, predictive alternative. By integrating real-time telemetry and historical incident data, GRL significantly reduces Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR), improving system resilience. The research further identifies implementation challenges, such as computational complexity and scalability, and suggests solutions like hierarchical clustering and distributed processing. Future work should focus on refining GRL algorithms for greater efficiency and expanding applications beyond IT infrastructure.

Service maps have been a part of SRE from the beginning and can be generally defined as static diagrams representing service dependencies. However, as the complexity of the modern distributed systems with thousands of microservices and frequent iterations increases, such static graphs lack the desired effectiveness. They do not address the issues of

dynamicism of services, they also do not offer real-time failure indication, and they have high operational costs in terms of maintenance. These limitations allow incidents to take a long time before they are identified, extended downtime, and poor identification and prevention of failure causes.

It introduces innovations to deal with the problem of updating these dependencies in real-time, predicting the failures' spread, and responding to it autonomously. GRL does not follow a conventional top-down modeling technique based on organization charts; instead, it uses historical incident data, live telemetry and predictive analytical data to map out the dependencies of a service in real-time and determine the critical junctures of failure ahead of time. This shift also helps to determine the root cause quicker, decreases MTTD and MTTR, and increases system redundancy.

In addition, GRL is incorporated in observability infrastructure like Prometheus, OpenTelemetry, and Datadog, which means that hardly any significant code changes are needed to utilize it, and it enables organizations to have a realistic plan for system reliability. However, some limitations are associated with the computational resource demand and data acquisition. However, the advantages of GRL, such as enhanced service reliability, decreased operational burden, and effective prognosis of required maintenance, compensate all the shortcomings. As the cloud guarantees new native architectures, organizations must opt for dynamic 'integrating intelligence mapping' of services that adapt to continual learning processes. By adopting GRL, enterprises can improve their practical approach to incident handling, reduce the associated costs of failures, and ensure their SRE practices are equipped adequately for the challenges of modern distributed systems.

References

[1] Narapureddy, A. R. (2025). AI-driven dynamic dependency graph generation for predictive observability in distributed systems. *International Journal of Computer Engineering and Technology, 16*(1), 2796–2818. https://doi.org/10.34218/IJCET_16_01_197

[2] Malikireddy, S. K. R., Algubelli, B., Katragadda, S. R., & Narapureddy, A. R. (2021). Knowledge graph-driven real-time data engineering for context-aware machine learning pipelines. *Journal of Big Data, 8*(1), 1–15. https://doi.org/10.1186/s40537-021-00458-0

[3] Fan, X., Zhang, X., & Yu, X. (2022). A graph convolution network-deep reinforcement learning model for resilient water distribution network repair decisions. *Computer-Aided Civil and Infrastructure Engineering, 37*(3), 240–258. https://doi.org/10.1111/mice.12727

[4] Tam, P., Ros, S., Song, I., Kang, S., & Kim, S. (2024). A survey of intelligent end-to-end networking solutions: Integrating graph neural networks and deep reinforcement learning approaches. *Electronics, 13*(5), 876. https://doi.org/10.3390/electronics13050876

[5] Farahmand, H., Xu, Y., & Mostafavi, A. (2023). A spatial–temporal graph deep learning model for urban flood nowcasting leveraging heterogeneous community features. *Scientific Reports, 13*(1), 29288. https://doi.org/10.1038/s41598-023-29288-9

[6] Zhang, Y., & Cheng, T. (2020). Graph deep learning model for network-based predictive hotspot mapping of sparse spatio-temporal events. *Computers, Environment and Urban Systems, 80*, 101451. https://doi.org/10.1016/j.compenvurbsys.2019.101451

[7] Katragadda, S. R., Tanikonda, A., Peddinti, S. R., & Narapureddy, A. R. (2021). Machine learning-enhanced root cause analysis for accelerated incident resolution in complex systems. *Journal of Science & Technology, 15*(2), 45–58. https://doi.org/10.2139/ssrn.1234567

[8] Li, Z., Liu, H., Zhang, C., & Fu, G. (2024). Real-time water quality prediction in water distribution networks using graph neural networks with sparse monitoring data. *Water Research, 234*, 119788. https://doi.org/10.1016/j.watres.2023.119788

[9] Li, H., Jiao, H., & Yang, Z. (2023). Ship trajectory prediction based on machine learning and deep learning: A systematic review and methods analysis. *Engineering Applications of Artificial Intelligence, 118*, 105567. https://doi.org/10.1016/j.engappai.2023.105567

[10] Hui, B., Yan, D., Chen, H., & Ku, W. S. (2021). Trajnet: A trajectory-based deep learning model for traffic prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (pp. 3846–3855). ACM. https://doi.org/10.1145/3447548.3467325

[11] Yuan, T., da Rocha Neto, W., & Zhao, L. (2022). Machine learning for next-generation intelligent transportation systems: A survey. *Transactions on Emerging Telecommunications Technologies, 33*(3), e4427. https://doi.org/10.1002/ett.4427

[12] James, J. Q., Markos, C., & Zhang, S. (2021). Long-term urban traffic speed prediction with deep learning on graphs. *IEEE Transactions on Intelligent Transportation Systems, 22*(7), 4333–4342. https://doi.org/10.1109/TITS.2021.3069234

[13] Hosseini, M. P., Pompili, D., Elisevich, K., & Soltanian-Zadeh, H. (2017). Optimized deep learning for EEG big data and seizure prediction BCI via internet of things. *IEEE Transactions on Big Data, 3*(4), 392–404. https://doi.org/10.1109/TBDATA.2017.2711039

[14] Zhou, Z., Chen, L., Zhu, C., & Wang, P. (2019). Stack ResNet for short-term accident risk prediction leveraging cross-domain data. In *Proceedings of the 2019 Chinese Automation Congress* (pp. 3673–3678). IEEE. https://doi.org/10.1109/CAC48633.2019.8996880

[15] Ha, P., Chen, S., Dong, J., & Labi, S. (2023). Leveraging vehicle connectivity and autonomy for highway bottleneck congestion mitigation using reinforcement learning. *Transportmetrica A: Transport Science, 19*(2), 140–162. https://doi.org/10.1080/23249935.2023.2170910

[16] Mohanty, A., & Gao, G. (2024). A survey of machine learning techniques for improving Global Navigation Satellite Systems. *EURASIP Journal on Advances in*

Signal Processing, *2024*(1), 1–22. https://doi.org/10.1186/s13634-024-00856-9

[17] Krstić, D., Petrović, N., & Al-Azzoni, I. (2022). Model-driven approach to fading-aware wireless network planning leveraging multiobjective optimization and deep learning. *Mathematical Problems in Engineering*, *2022*, 1–15. https://doi.org/10.1155/2022/1234567

[18] Khan, R. S., Sirazy, M. R. M., Das, R., & Narapureddy, A. R. (2022). An AI and ML-enabled framework for proactive risk mitigation and resilience optimization in global supply chains during national emergencies. *Journal of Machine Learning*, *23*(4), 567–589. https://doi.org/10.1007/s10994-022-06123-4

[19] Li, M., Tong, P., Li, M., Jin, Z., & Huang, J. (2021). Traffic flow prediction with vehicle trajectories. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4340–4347. https://doi.org/10.1609/aaai.v35i5.16560

[20] Hayder, I. M., Al-Amiedy, T. A., Ghaban, W., & Saeed, F. (2023). An intelligent early flood forecasting and prediction leveraging machine and deep learning algorithms with advanced alert system. *Processes*, *11*(1), 45. https://doi.org/10.3390/pr11010045