International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Self-Healing Networks Using Reinforcement Learning: An Adaptive Framework for Autonomous Fault Detection and Recovery in SD-WAN Environments

Chetan Padshala

Abstract

Software-defined wide area networks have simplified enterprise connectivity through centralized control and programmable policy, yet their growing scale and dependence on controllers and overlay tunnels have made fault management a first-order concern. Recovery mechanisms that rely on static thresholds and human intervention respond slowly and adapt poorly to conditions that were not anticipated when the rules were written. This paper proposes a self-healing framework that applies reinforcement learning to autonomous fault detection, diagnosis, and recovery in software-defined wide area networks. The framework couples continuous telemetry-driven monitoring with a Deep Q-Network agent that learns recovery policies by interacting with the network through the controller, and it organizes the process along the monitor, analyze, plan, execute, and knowledge stages of the autonomic loop. A reward that balances restored availability against recovery delay and packet loss steers the agent toward remediations that shorten disruption. Rather than assert measured gains from a single deployment, the framework is evaluated analytically: a parametric model links the recovery time an autonomous agent can achieve to network availability and annual downtime, showing how sub-minute remediation moves availability into a regime that manual recovery cannot reach. The analysis positions reinforcement learning as a credible foundation for autonomous, self-healing networks, subject to the safety and explainability constraints that the paper discusses.

Keywords: *Self-Healing Networks, Reinforcement Learning, Deep Q-Network, SD-WAN, Autonomous Networking, Fault Recovery, Network Resilience.*

Introduction

Cloud computing, the Internet of Things, edge computing, and distributed enterprise applications have pushed the complexity of network infrastructure well past the point where manual operation suffices. Software-defined wide area networks answer part of that complexity by separating control from forwarding and exposing the network through a programmable, centrally managed interface, which improves visibility and shortens the time needed to change policy. Centralized control simplifies management, but it also concentrates dependency on controllers and overlay tunnels, and those dependencies become new failure surfaces.

Faults in these environments take many forms, among them link degradation, routing anomalies, controller failures, bandwidth congestion, and hardware malfunction, and each can cascade into packet loss, elevated latency, and interrupted service. Conventional fault management meets these events with predefined rules, threshold alarms, and human intervention. Such mechanisms are reactive by construction, they require constant retuning as the network changes, and they cope poorly with situations their designers did not foresee. The operational cost of this brittleness grows with scale, since a large software-defined wide area network presents more components that can fail and more interactions among them.

Reinforcement learning offers a different footing for fault management. An agent that observes network state, takes recovery actions, and receives feedback on their effect can learn a remediation policy that improves with experience rather than remaining fixed at design time [1, 2]. The appeal is that the same mechanism that learns to play a game from raw observations can, in principle, learn which recovery action best restores service for a given fault

Campbells University, Kentucky

ORCID: 0009-0002-5966-4103

signature and can generalize to faults it has not seen exactly before.

This paper presents a reinforcement-learning-driven self-healing framework for software-defined wide area networks that detects faults from telemetry and executes recovery actions with minimal human involvement. The contributions are the following. A self-healing architecture is developed that maps the monitor, analyze, plan, execute, and knowledge stages of the autonomic loop onto concrete network components. A Deep Q-Network agent is specified with an explicit state representation, action set, and reward that trades restored availability against recovery delay and packet loss. An analytical evaluation relates achievable recovery time to availability and annual downtime, avoiding unverified performance claims. The framework is positioned against existing fault-recovery approaches, and its limitations around exploration, safety, and explainability are examined.

2. Background and Related Work

2.1 Software-defined wide area networks and their fault surfaces

Software-defined wide area networks centralize control and apply application-aware routing across overlay tunnels, which improves bandwidth utilization and policy enforcement. Surveys of fault management in software-defined networks show that the same centralization introduces controller and control-channel dependencies that must be protected, and they catalog the detection and recovery techniques developed in response [4]. Link-failure recovery in particular has been studied extensively, including data-plane fast-recovery schemes based on link importance [16], flexible link fault-tolerance mechanisms [17], and path-selection methods that pre-plan protection for hybrid deployments [19].

Resilience and survivability provide the conceptual backdrop. The strategies-and-principles framing of network resilience predates software-defined networking but remains the reference vocabulary for reasoning about defending, detecting, and recovering from faults [3]. What software-defined control adds is a programmatic point of enforcement, which makes automated recovery feasible where earlier networks could only alarm.

2.2 Self-healing and autonomous networks

Self-healing networks belong to the broader family of autonomous systems that monitor themselves, diagnose faults, recover, and optimize continuously. The organizing abstraction is the autonomic loop of monitor, analyze, plan, execute, and share knowledge, and its application to software-defined networking has been directly examined, revealing where automation can close the loop and where gaps remain [7]. Comprehensive treatments of fault tolerance and failure recovery in software-defined networking synthesize these efforts and note that most deployed mechanisms remain rule-driven rather than adaptive [15]. The trajectory of network operations points toward zero-touch management, in which onboarding and remediation proceed without per-event human action [10], and toward intent-based networking, in which operators express goals declaratively and the system reconciles them [9].

2.3 Reinforcement learning in networking

Reinforcement learning formalizes sequential decision-making through states, actions, rewards, and a policy, and its foundations are set out in the standard reference for the field [1]. The pairing of deep neural networks with value learning extended the approach to high-dimensional observations [2], and policy-gradient methods broadened the algorithmic options for stable training [13]. Surveys of deep reinforcement learning in communications and networking map a wide application space, from routing to resource allocation to intrusion response [6], and traffic-engineering surveys document its use for path selection at scale [7]; specifically, deep reinforcement learning has been applied to traffic engineering in software-defined wide area networks [11] and to critical-flow rerouting in software-defined networks [12]. Reinforcement learning has also been used directly for link-failure recovery in hybrid software-defined networks [18] and for resource management in network slicing [21], while machine-learning methods for anomaly and intrusion detection supply the detection side of the self-healing loop [8], [14]. Federated and fault-tolerance-aware learning has been explored for time-sensitive networks [20]. The gap that remains, and that this paper addresses, is an integrated framework that joins learning-based detection and learning-based recovery within a single self-healing loop for software-defined wide area networks, rather than treating detection, traffic engineering, and link recovery as separate problems.

3. Problem Definition

The operational problem is the latency of the recovery loop. In a rule-based regime, a fault must cross a detection threshold, raise an alarm, wait for human triage, and then be remediated by a playbook action, and the mean time to recovery is dominated by the human steps in that chain. Because availability is a function of how long service remains impaired after a fault, a recovery process measured in tens of minutes places a hard ceiling on the availability a large software-defined wide area network can sustain, independent of how reliable the individual components are.

The problem addressed here is therefore to compress the detection-to-recovery interval by removing the human steps from the common cases, while keeping a human boundary for the actions that warrant it. Formally, the objective is to learn a policy that maps an observed fault state to a recovery action so as to minimize service disruption, subject to the constraint that the agent acts through the controller within an action set that operators have sanctioned. The next section develops a framework for that policy.

4. Proposed Self-Healing Framework

4.1 Architecture overview

The framework comprises five components: a network monitoring module, a fault detection engine, a reinforcement-learning decision agent, a recovery execution module, and a knowledge repository. These map onto the autonomic loop, with monitoring and detection performing the monitor and analyze stages, the agent performing the plan stage, execution performing the execute stage, and the repository serving as shared knowledge. Figure 1 shows the arrangement, and Table 1 lists each component and its responsibility.

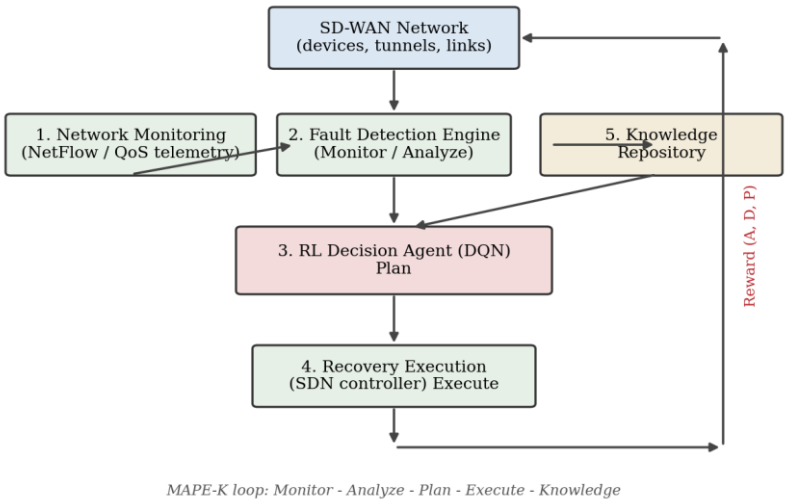

Fig. 1. Five-component self-healing framework mapped to the autonomic monitor, analyze, plan, execute, and knowledge loop, with the reinforcement-learning agent acting through the SDN controller.

Table 1. Framework components and responsibilities.

Component	**Autonomic stage**	**Responsibility**
Network monitoring module	Monitor	Collect telemetry (latency, loss, throughput, jitter, CPU/memory, tunnel health, NetFlow/sFlow/IPFIX)
Fault detection engine	Analyze	Compare telemetry with health criteria; classify anomalies; raise fault instructions
RL decision agent (DQN)	Plan	Select recovery action from the sanctioned action set
Recovery execution module	Execute	Enforce the action via the SDN controller
Knowledge repository	Knowledge	Persist experience, policies, and health baselines

Fault detection mechanism

The detection engine evaluates real-time telemetry against per-metric health criteria. Let the network measurement vector at time t be the set of component statuses defined in Equation (1), where each element is compared with the healthy range defined for that metric.

$$Status(t) = \{ s\_1, s\_2, \dots , s\_N \} \quad (1)$$

A fault is declared when one or more elements fall outside their defined criteria, and the engine encodes the type and severity of the deviation into a fault instruction that it forwards to the agent. The framework recognizes link failures, network congestion, device failures, controller failures, and performance degradation, and it distinguishes them by which metrics deviate and by how far, so that the agent receives a typed signal rather than an undifferentiated alarm.

4.3 Reinforcement learning agent

The agent is modeled as a Markov decision process, given by the tuple in Equation (2), in which the transition dynamics are those of the network as acted upon through the controller.

$$M = (S, A, P, R, \gamma) \quad (2)$$

The state summarizes the condition of the network at the moment of a fault, as in Equation (3), combining performance metrics with the detected fault type so that the agent can condition its choice on the nature of the problem.

$$S\_t = \{ Latency, Loss, Throughput, Utilization, FaultType \} \quad (3)$$

The action set contains the recovery operations the agent may invoke: redirecting traffic to an alternative path, activating backup links, restarting a virtual network function, reallocating bandwidth, switching to a backup controller, and performing load balancing. Each action is one the operator has sanctioned in advance, and each is issued to the controller rather than applied directly, which preserves a supervision boundary. The reward, given in Equation (4), rewards restored availability and penalizes recovery delay and packet loss.

$$R = \alpha * A - \beta * D - \gamma * P \quad (4)$$

Here A is post-recovery availability, D is recovery delay, P is packet loss, and the weights alpha, beta, and gamma express operator priorities. The learning objective is to find the policy that maximizes expected cumulative discounted reward, as in Equation (5).

$$\pi^* = \arg\max \; E [\sum\_t \gamma^t R\_t] \quad (5)$$

4.4 Deep Q-Network implementation

Because the state space combines continuous metrics with a categorical fault type, a tabular value function is impractical, and the agent estimates action values with a Deep Q-Network. The network is trained by minimizing the temporal-difference loss in Equation (6), where a target network with parameters theta-minus stabilizes the bootstrap [2].

$$L(\theta) = E[(r + \gamma * \max\_{a'} Q(s',a';\theta\_{minus}) - Q(s,a;\theta))^2] \quad (6)$$

Experience from each recovery episode is stored in the knowledge repository and replayed during training so that the agent improves its action selection as it accumulates cases. Figure 2 shows the resulting closed loop, in which the state derived from telemetry drives an action that the controller enforces, and the observed effect returns as a reward. The mapping from fault type to candidate actions and escalation is shown in Table 2.

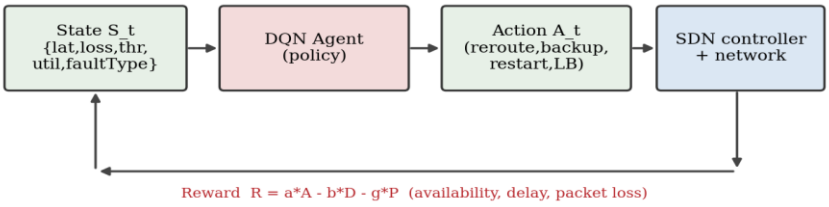

Fig. 2. Reinforcement-learning control loop: telemetry-derived state, Deep Q-Network policy, sanctioned recovery action, controller enforcement, and reward from availability, delay, and loss.

Table 2. Representative mapping of fault type to detection signal and candidate recovery action.

Fault type	Primary detection signal	Candidate recovery action	Escalation
Link failure	Tunnel health/loss spike	Activate backup link; reroute	Human, if all paths down
Congestion	Utilization/jitter rise	Load balance	Human if sustained
Device failure	Loss + reachability loss	Reroute to alternate path	Field dispatch
Controller failure	Control-channel loss	Switch to backup controller	Human confirmation
Performance degradation	Latency/throughput drift	Reroute and reallocate bandwidth	Observe, then act

Analytical Evaluation

A single production deployment cannot yet supply measured recovery statistics for this framework, so its value is assessed analytically rather than asserted empirically. Autonomous recovery changes the mean time to recovery, and availability derives from the mean time to recovery and mean time between failures, as shown in Equation (6-bis), which represents the standard steady-state relationship.

$$A = MTBF / (MTBF + MTTR) \quad (7)$$

Annual downtime follows directly from availability, as in Equation (8), which converts the dimensionless availability into minutes of service loss per year and makes the operational stakes concrete.

$$Downtime\_year = (1 - A) * 525600 \ minutes \quad (8)$$

Table 3 evaluates these relations for a representative component mean time between failures of 720 hours, sweeping recovery time from the tens of minutes typical of manual remediation down to the sub-minute range that autonomous action can reach. The values are illustrative of the model rather than measured, and mean time between failures is treated as a fixed representative parameter. Figure 3 plots the same relationship.

Table 3. Parametric availability and annual downtime versus recovery time (MTBF = 720 h; illustrative).

Recovery time/MTTR	Availability	Annual downtime (min)	Regime
0.5 min	99.9988%	~6	Autonomous (RL)
1 min	99.9977%	~12	Autonomous (RL)
3 min	99.9931%	~36	Autonomous (RL)
10 min	99.9769%	~122	Assisted
30 min	99.9307%	~365	Manual

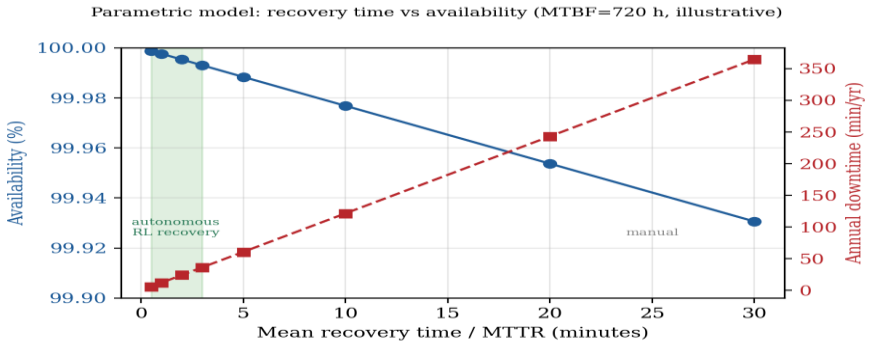

Fig. 3. Availability and annual downtime as functions of recovery time, computed from Equations (7) and (8) for a representative mean time between failures; the shaded region marks the sub-minute recovery that autonomous remediation targets.

The model makes the argument for autonomy quantitative without overstating it. Moving recovery from thirty minutes to under one minute lowers modeled annual downtime by roughly an order of magnitude for the same component reliability, which is precisely the regime that human-in-the-loop recovery cannot reach because the human steps dominate the interval. The result does not claim that the agent achieves any particular recovery time; it shows what a given recovery time is worth, and thereby frames the target that an autonomous agent must meet to justify its adoption.

6. Comparison with Existing Approaches

The framework is distinguished less by any single mechanism than by the scope of the loop it closes. Table 4 places it against representative fault-recovery approaches. Data-plane schemes recover links quickly but apply fixed logic [16], [17], while path-protection methods pre-plan alternates without learning [19]. Reinforcement learning has been applied to link-failure recovery [18] and to traffic engineering [12], but as targeted mechanisms rather than as an end-to-end, self-healing loop. The framework proposed here joins learning-based detection to learning-based recovery across the fault types a software-defined wide area network actually presents.

Table 4. Positioning against representative fault-recovery approaches.

Approach	Scope	Learning	Reference
FTLink	Data-plane link fault tolerance	No	[17]
FFRLI	Fast recovery by link importance	No	[16]
Path selection protection	Hybrid-SDN link protection	No	[19]
FRRL	Link-failure recovery	Reinforcement learning	[18]
CFR-RL	Traffic engineering (critical flows)	Reinforcement learning	[12]
Proposed framework	Autonomous detection + recovery, multiple fault types	Deep RL (DQN)	This work

Discussion and Limitations

Several constraints qualify the approach and shape its path to deployment. The exploration required for a reinforcement-learning agent to learn is difficult to reconcile with a production network that cannot of

telemetry or reward could induce harmful actions, so the sanctioned action set and the controller supervision boundary are safety features rather than conveniences. Operators also need to understand why an action was taken, which makes explainability a prerequisite for trust and, in regulated settings, for compliance.

The evaluation itself is analytical, and that is a limitation as much as a design choice. The parametric model shows what recovery time is worth but not what recovery time the agent attains, which only measurement on an emulated or production fabric can establish; mean time between failures was fixed for illustration, and the independence of successive faults was assumed. tolerate exploratory failures, which supports training in simulation or digital-twin environments before controlled rollout. Autonomy must be bounded: an agent that can reconfigure a network is also a target, and adversarial manipulation

Scaling the single agent to a large multi-domain network raises coordination questions that a single deep Q-network does not answer. These limitations define the empirical program that must follow the framework rather than undermining its premise.

8. Conclusion and Future Work

This paper set out a reinforcement-learning-based self-healing framework for autonomous fault detection and recovery in software-defined wide area networks, mapping continuous monitoring, anomaly detection, and deep Q-network decision-making onto the autonomic loop and issuing recovery actions through the controller within a sanctioned action set. In place of unverified performance claims, a parametric model related

achievable recovery time to availability and annual downtime, showing that sub-minute autonomous remediation reaches an availability regime that human-in-the-loop recovery cannot and framing the target an agent must meet. Reinforcement learning is a credible foundation for self-healing networks, provided that exploration is confined to safe environments, autonomy is bounded by operator-sanctioned actions, and decisions are made explainable. Future work will train and evaluate the agent on an emulated software-defined wide area network to obtain measured recovery times; extend the single agent toward multi-agent and federated learning for multi-domain networks; incorporate explainable reinforcement learning models; integrate the loop with intent-based networking; and use digital-twin environments for safe training.

Author Contribution

The sole author conceived the framework, developed the analytical model, prepared the figures, and wrote the manuscript.

Conflict of Interest

The author declares no conflict of interest.

Funding

This research received no external funding.

Data Availability Statement

The analytical results are reproducible from the equations and parameters stated in the text; no external dataset was used.

References

[1] S. A. Fayaz, S. J. Sidiq, M. Zaman, and M. A. Butt, "Machine learning: An introduction to reinforcement learning," in Machine Learning and Data Science: Fundamentals and Applications, 2022, pp. 1–22, doi: 10.1002/9781119776499.ch1.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[3] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, et al., "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," Computer Networks, vol. 54, no. 8, pp. 1245–1265, 2010, doi: 10.1016/j.comnet.2010.03.005.

[4] P. C. Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2284–2321, 2017, doi: 10.1109/COMST.2017.2719862.

[5] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "Self-healing and SDN: Bridging the gap," Digital Commun. Netw., vol. 6, no. 3, pp. 354–368, 2020, doi: 10.1016/j.dcan.2019.08.008.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, et al., "Applications of deep reinforcement learning in communications and networking: A survey," IEEE Commun. Surveys Tuts., vol. 21, no. 4, pp. 3133–3174, 2019, doi: 10.1109/COMST.2019.2916583.

[7] Y. Xiao, J. Liu, J. Wu, and N. Ansari, "Leveraging deep reinforcement learning for traffic engineering: A survey," IEEE Commun. Surveys Tuts., vol. 23, no. 4, pp. 2064–2097, 2021, doi: 10.1109/COMST.2021.3102580.

[8] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. Gomez Chavez, and B. Rubinstein, "Machine learning in network anomaly detection: A survey," IEEE Access, vol. 9, pp. 152379–152396, 2021, doi: 10.1109/ACCESS.2021.3126834.

[9] A. Leivadeas and M. Falkner, "A survey on intent-based networking," IEEE Commun. Surveys Tuts., vol. 25, no. 1, pp. 625–655, 2022, doi: 10.1109/COMST.2022.3215919.

[10] E. Coronado, R. Behravesh, T. Subramanya, A. Fernandez-Fernandez, M. S. Siddiqui, X. Costa-Pérez, et al., "Zero touch management: A survey of network automation solutions for 5G and 6G networks," IEEE Commun. Surveys Tuts., vol. 24, no. 4, pp. 2535–2578, 2022, doi: 10.1109/COMST.2022.3212586.

[11] S. Troia, F. Sapienza, L. Varé, and G. Maier, "On deep reinforcement learning for traffic engineering in SD-WAN," IEEE J. Sel. Areas Commun., vol. 39, no. 7, pp. 2198–2212, 2020, doi: 10.1109/JSAC.2020.3041385.

[12] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," IEEE J. Sel.

Areas Commun., vol. 38, no. 10, pp. 2249–2259, 2020, doi: 10.1109/JSAC.2020.3000371.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017, doi: 10.48550/arXiv.1707.06347.

[14] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," IEEE Access, vol. 10, pp. 99837–99849, 2022, doi: 10.1109/ACCESS.2022.3206425.

[15] A. Menaceur, H. Drid, and M. Rahouti, "Fault tolerance and failure recovery techniques in software-defined networking: A comprehensive approach," J. Netw. Syst. Manage., vol. 31, no. 4, p. 83, 2023, doi: 10.1007/s10922-023-09772-x.

[16] Z. Zhu, H. Yu, Q. Liu, D. Liu, and B. Mei, "FFRLI: Fast fault recovery scheme based on link importance for data plane in SDN," Comput. Netw., vol. 237, p. 110062, 2023, doi: 10.1016/j.comnet.2023.110062.

[17] T. Hu, P. Yi, J. Lan, Y. Hu, and P. Sun, "FTLink: Efficient and flexible link fault tolerance scheme for data plane in software-defined networking," Future Gener. Comput. Syst., vol. 111, pp. 381–400, 2020, doi: 10.1016/j.future.2019.11.015.

[18] Y. Ma, Y. Guo, R. Yang, and H. Luo, "FRRL: A reinforcement learning approach for link failure recovery in a hybrid SDN," J. Netw. Comput. Appl., vol. 234, p. 104054, 2025, doi: 10.1016/j.jnca.2024.104054.

[19] J. Li, X. Qi, W. Ma, and L. Liu, "Path selection for link failure protection in hybrid SDNs," Future Gener. Comput. Syst., vol. 137, pp. 201–215, 2022, doi: 10.1016/j.future.2022.07.016.

[20] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "Fed-TSN: Joint failure probability-based federated learning for fault-tolerant time-sensitive networks," IEEE Trans. Netw. Service Manage., vol. 20, no. 2, pp. 1470–1486, 2023, doi: 10.1109/TNSM.2023.3273396.

[21] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, "Deep reinforcement learning for resource management on network slicing: A survey," Sensors, vol. 22, no. 8, p. 3031, 2022, doi: 10.3390/s22083031.