Intelligent Traffic Steering Using Reinforcement Learning

Hakan Bektas
Faculty of Science
University of Amsterdam
hakan.bektas2@student.uva.nl

Marios Avgeris
Faculty of Science
University of Amsterdam
m.avgeris@uva.nl

***Abstract*—Traditional IP routing relies on algorithms such as OSPF and BGP, which use predefined metrics and static decision-making processes to construct forwarding tables based on the longest prefix match. While these approaches are effective in stable network environments, they often fail to adapt efficiently to dynamic changes in network topology, leading to suboptimal routing decisions, increased latency, and packet loss. In this paper, we explore the use of reinforcement learning (RL) for intelligent flow steering. RL enables smart routers or programmable switches to dynamically adapt to changing conditions by learning optimal paths through iterative feedback mechanisms involving rewards and penalties. Our RL-based method demonstrates improved adaptability and efficiency in path selection with optimized metrics, providing a significant advancement over traditional routing protocols. This paper presents the methodology, experimental results, and a discussion on the implications of RL-driven path optimization.**

Keywords: Path Selection, Reinforcement Learning, Network Optimization, INT (In-band Network Telemetry)

I. Introduction

The rapid expansion of modern networks, fueled by the increasing demand for low-latency applications, cloud computing, and real-time communication, has posed significant challenges for traditional IP routing mechanisms. Conventional routing protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) determine paths based on predefined static or semi-dynamic metrics, typically optimizing for shortest paths or link cost. While these approaches work well under stable conditions, they struggle to adapt to real-time network variations such as congestion, link failures, or dynamic traffic patterns [1]. As networks continue to grow in size and complexity, the need for more intelligent and adaptive routing strategies becomes apparent.

A key limitation of traditional routing protocols is their lack of **real-time adaptability**. Static metrics do not always capture transient network fluctuations, leading to suboptimal routing decisions that may increase packet loss, latency, or bandwidth underutilization. Even with dynamic routing protocols, updates often occur at fixed intervals or rely on threshold-based triggers, which can lead to delayed responses to sudden network changes. Moreover, conventional routing decisions are typically decentralized and limited to individual routers' perspectives, preventing a more holistic optimization of the network.

Recent advancements in **programmability and artificial intelligence (AI)** offer new opportunities for overcoming these challenges [2]. Software-Defined Networking (SDN) allows for centralized control over routing policies, enabling real-time adjustments based on telemetry data [3]. Additionally, **programmable networks** and **P4-based data plane programmability** provide further flexibility in traffic management by allowing custom packet processing at line rate. These technologies enable real-time decision-making and the integration of reinforcement learning directly within the network fabric [2]. Furthermore, reinforcement learning (RL) has emerged as a powerful approach for decision-making in dynamic environments. By leveraging RL, routers can continuously learn and adjust their routing decisions based on observed network conditions, rewarding efficient paths and penalizing suboptimal ones.

Specifically in this work, we propose an **RL-based adaptive routing mechanism** that dynamically selects paths based on real-time network telemetry, with a primary focus on **latency minimization**. Our contributions are summarized as follows:

- We design a reinforcement learning framework for routing decisions, where a **Smart Router** learns optimal path selection based on **network telemetry**.
- We introduce an experimental setup that simulates real-time network variations by modifying network conditions, demonstrating how an RL-based router can adapt.
- We evaluate the impact of inference and learning phases on routing stability, highlighting the effectiveness of adaptive decision-making over fixed-metric routing.

The remainder of this paper is structured as follows. Section II provides an overview of related work in RL-based routing. Section III introduces the system model, including the network topology and delay modeling. Experimental results and their implications on network performance are presented in Section IV. Finally, Section V concludes the paper and highlights future research directions.

II. Related Works

Recent research in intelligent traffic routing and reinforcement learning (RL) has provided various methodologies to optimize network performance. This section presents an overview of key contributions in the field, focusing on reinforcement learning-based routing, congestion control, and adaptive path selection.

Intelligent traffic routing has been explored extensively to optimize network performance in various environments. One such study introduces three primary metrics for controlling packet congestion: the packet set variation, traffic speed, and congestion index [4]. The proposed methodology calculates variations between actual and predicted congestion, aiming to improve routing decisions by dynamically adjusting congestion thresholds instead of relying on static values. These metrics are used to enhance network efficiency by dynamically adapting routing choices based on real-time congestion measurements.

Packet loss is a critical factor in routing optimization, and various studies have investigated strategies to mitigate its impact. A probability-based approach has been formulated to measure packet loss across bidirectional transmission paths, allowing for more accurate estimation of packet loss rates [5]. By incorporating these probabilities into routing decisions, the model effectively reduces packet retransmissions and enhances network reliability. The study highlights the need for packet loss-aware routing protocols that dynamically adjust based on real-time network conditions.

Active Queue Management (AQM) has also been enhanced through reinforcement learning-based techniques. The DE-SiRED framework, a P4-based deep reinforcement learning (DRL) approach, leverages in-band network telemetry to adjust queue parameters dynamically [3]. This approach optimizes packet scheduling based on real-time network conditions, significantly improving queue stability and mitigating congestion. The study demonstrates that reinforcement learning can dynamically adjust queue thresholds to ensure low packet drop rates while maintaining high network efficiency.

Another domain of interest is the application of reinforcement learning for optimizing virtual network function (VNF) placement. Using Stochastic Learning Automata (SLA), a framework has been proposed to enable self-adaptive VNF placement, enhancing service availability and resilience [6]. The approach is designed to improve service function chaining (SFC) by dynamically adjusting network paths based on observed traffic patterns and system failures. The research highlights the potential of reinforcement learning in optimizing distributed network functions, ensuring efficient load balancing, and reducing latency in edge-cloud computing environments.

Path selection mechanisms in dynamic networks have also been investigated, particularly in the context of **Software-Defined Wide Area Networks (SD-WANs)**. A deep reinforcement learning-based approach has been proposed for robust path selection, emphasizing the limitations of traditional shortest-path routing algorithms [7]. The study highlights how reinforcement learning-based path selection outperforms conventional routing algorithms in terms of stability and adaptability, making it a viable solution for large-scale, high-variability networks.

These studies collectively demonstrate the effectiveness of reinforcement learning and programmable networking in tackling various challenges related to traffic routing, congestion control, VNF placement, and path selection. The adoption of **RL-based approaches, network telemetry, and SDN programmability** is expected to play a crucial role in the future of network optimization.

Building on advancements in reinforcement learning for adaptive path selection, this work focuses on leveraging network telemetry, specifically delay variations, to dynamically steer traffic through a **Smart Router**. While existing research has optimized routing based on congestion metrics, packet loss, or queue management, the proposed approach integrates reinforcement learning within a real-time adaptive framework to enhance responsiveness and efficiency in routing decisions. By utilizing RL-driven smart routers, dynamic telemetry feedback is incorporated to ensure adaptive traffic management, addressing the challenges posed by delay-sensitive applications in modern networks.

III. Network Topology and Experiment Design

A. Topology

The design involves a network with five main entities:

- **Server A:** Generates traffic and sends it to the Smart Router, with Server B as the end destination.
- **Smart Router:** An SDN-capable or programmable switch that routes traffic to either Router 1 or Router 2 based on dynamic probabilities.
- **Router 1 and Router 2:** Forward data to Server B.
- **Server B:** Computes metrics and sends feedback to the Smart Router.

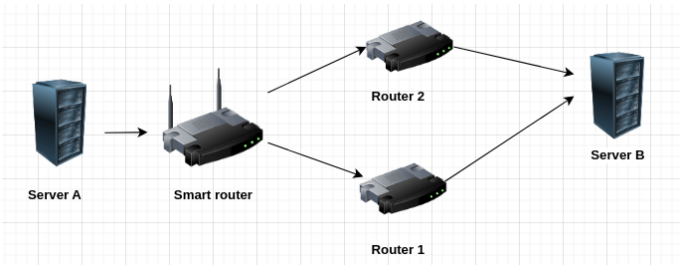

Fig. 1. Updated simulation design

The goal here is to minimize packet delay by learning which of the two available routers provides the fastest transmission time under changing network conditions.

B. Experiment Parameters and Iteration Process

The experiment runs for a total of 80 iterations. It consists of alternating **learning** and **inference** phases:

- **Learning Phase:** The Smart Router dynamically updates its routing probabilities based on observed delays. Learning continues until convergence criteria are met.
- **Inference Phase:** The system switches to a fixed decision-making process using the last learned probabilities. This prevents fluctuations while applying the learned routing behavior.

The Smart Router selects a path based on a reinforcement learning process, adjusting routing probabilities through network telemetry feedback. A reward-based update mechanism

is applied in each iteration to steer the learning process. The key parameters used in the simulation are:

- **Total iterations:** 80
- **Learning iterations before checking convergence:** 29
- **Convergence threshold:** 0.05
- **Consecutive stable updates required for convergence:** 3
- **Inference iterations:** 15
- **Base propagation delay for Router 1:** 0.95 ms
- **Base propagation delay for Router 2:** 0.2 ms
- **Injected delay on Router 2:** +2.3 ms (from iteration 20 to 40) and +0.5 ms (from iteration 60 to 70)

The simulation consists of two main phases: a **learning phase** and an **inference phase**. The Smart Router dynamically updates routing probabilities during the learning phase based on observed delays, while in the inference phase, it applies the last learned probabilities without further adjustments. The total number of iterations is set to 80, ensuring sufficient time for adaptation. Learning iterations are capped at 29 before checking for convergence, which is determined by a probability change threshold of 0.05. Convergence is confirmed if the probability updates remain stable for at least three consecutive iterations. The system then transitions into the inference phase, where routing decisions remain fixed for 15 iterations before the next learning cycle begins. Router 1 and Router 2 have initial base propagation delays of 0.95 ms and 0.2 ms, respectively, making Router 2 the preferred choice at the start. However, additional overhead delay is introduced to Router 2 between iterations 20 and 40 (+2.3 ms), forcing the Smart Router to shift traffic towards Router 1. A smaller delay increase of +0.5 ms is added between iterations 60 and 70 to test how the system responds to minor disruptions. These parameters collectively define the learning process and ensure adaptive routing decisions based on real-time network conditions.

C. Iteration Breakdown

Each iteration consists of the following steps:

1) The Smart Router chooses either Router 1 or Router 2 based on current routing probabilities.
2) The packet experiences the corresponding delay for the chosen router.
3) A reward function computes the effectiveness of the choice based on delay reduction.
4) The routing probabilities are updated using reinforcement learning during the learning phase.
5) If the convergence threshold is reached, the system transitions to the inference phase, applying the learned routing decisions without further updates.

This setup ensures that the system continuously adapts to network conditions while stabilizing routing decisions over time.

D. Delay Modeling

The primary performance metric used in this model is the **end-to-end delay** experienced by packets traveling from **Server A** to **Server B**.Each path has a **propagation delay**, and the path through Router 2 initially has a lower delay, making it the preferred choice. However, at specific time intervals, we assume that the delay on one of the paths increases, leading the Smart Router to adjust its preference dynamically. Once the delay is reduced, the learning mechanism allows the system to re-optimize and favor the lower-delay path again. The total delay for a given path is given by:

$$D_{\text{path}} = D_{\text{base}} + D_{\text{overhead}}$$

where:

- D_{base} represents the inherent propagation delay of the path,
- D_{overhead} accounts for additional delay variations occurring during specific intervals.

$$D_{\text{path}} = D_{\text{base}} + D_{\text{overhead}}$$

where:

- D_{base} represents the inherent propagation delay of the path,
- D_{overhead} accounts for additional delay variations occurring during specific intervals.

E. Probability Update Mechanism

The Smart Router is implemented as an Automaton, following the reinforcement learning-based modeling of the Stochastic Learning Automata (SLA) algorithm [6]. It dynamically adjusts the probabilities of selecting a path based on observed delay values. The probability update follows the rule:

$$P_{\text{new}} = P_{\text{current}} + \alpha \times (\text{R} - P_{\text{current}})$$

where:

- α is the learning rate.
- Reward is a normalized score representing how favorable the observed delay was. The reward function assigns higher values to paths with lower delays, encouraging the Smart Router to prefer those routes.

The probability values are constrained within the range $[0, 1]$ to ensure a valid probability distribution.

Example:

- Suppose the current probability for steering the traffic to path 1 is $P_{\text{current}} = 0.6$.
- The computed reward for Router 1 is Reward $= 0.8$,
- The learning rate is $\alpha = 0.1$.

The updated probability becomes:

$$P_{\text{new}} = 0.6 + 0.1 \times (0.8 - 0.6) = 0.62$$

F. Reward Calculation and Decision Process

To determine how favorable a router is, we normalize delay into a **reward** in the range $[0, 1]$, where higher values indicate better performance. The reward is calculated as:

$$R = 1 - \frac{D_{\text{router}}}{D_{\max}}$$

where:

- D_{router} is the measured delay of the chosen router,
- $D_{\max}$ is a normalization factor representing the highest expected delay. It is determined experimentally.

Example:

- If the maximum expected delay is $D_{\max} = 2.0$,
- Router 1 has a delay of 1.2, its reward is:

$$R = 1 - \frac{1.2}{2.0} = 0.4$$

- Router 2 (before extra overhead) has a delay of 0.7, its reward is:

$$R = 1 - \frac{0.7}{2.0} = 0.65$$

- During the interval where Router 2 has an additional delay (making $D = 1.7$), its reward drops to:

$$R = 1 - \frac{1.7}{2.0} = 0.15$$

G. SLA-Based Routing Decision Algorithm

Algorithm 1 SLA-Based Routing Decision Algorithm

Initialize routing probabilities $P(R_1) = P(R_2) = 0.5$
Set learning rate α and convergence threshold ε
for each iteration t **do**
  Choose router a_t based on $P(R_1)$ and $P(R_2)$
  Measure delay D_{router}
  Compute reward:

$$R = 1 - \frac{D_{\text{router}}}{D_{\max}}$$

  Update probabilities independently:

$$P_{\text{new}}(R_1) = P_{\text{current}}(R_1) + \alpha \times (R - P_{\text{current}}(R_1))$$

$$P_{\text{new}}(R_2) = P_{\text{current}}(R_2) + \alpha \times (R - P_{\text{current}}(R_2))$$

  Check for convergence:
  if $|P_{\text{new}}(R_1) - P_{\text{current}}(R_1)| < \varepsilon$ for N iterations **then**
    Switch to inference mode
  end if
end for

IV. Results and Discussion

The simulations were conducted using Python 3.11.8 in Visual Studio Code (VSCode) 1.95.2 on a personal laptop running Ubuntu 22.04.1. The hardware specifications of the machine used for the experiments are as follows:

- **Processor:** 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (4 cores, 8 threads, max 4.7 GHz)
- **Memory (RAM):** 16 GB DDR4
- **Graphics:**
  - Integrated: Intel Iris Xe Graphics
  - Dedicated: NVIDIA GeForce MX450
- **Operating System:** Ubuntu 22.04.1 (Linux Kernel 6.8.0-51)
- **Virtualization Support:** Intel VT-x enabled

The reinforcement learning-based routing simulations were executed in Python, utilizing built-in libraries such as `random` and `matplotlib` for probability-based decision-making and visualization.

A. Simulation Setup

The simulation environment consists of a Smart Router selecting between two forwarding paths, each leading through Router 1 or Router 2, based on delay measurements. The key simulation parameters are:

- **Learning rate:** $\alpha = 0.1$
- **Number of iterations:** 80
- **Convergence threshold:** $\varepsilon = 0.05$
- **Inference iterations:** 15
- **Delay adjustments:** From iteration 20 to 40, the delay on one path is increased, and from iteration 60 to 70, a smaller delay is introduced.

The simulation consists of alternating learning and inference phases. During the learning phase, routing probabilities are updated dynamically based on observed delay metrics. When the system converges, it switches to inference mode, where it applies the last computed probabilities without further adjustments.

B. Experiments

We conduct two main experiments:

1) **Adaptive Routing (SLA-Based)**: The Smart Router dynamically updates probabilities based on delay feedback.
2) **Naive Routing (Fixed Choice)**: The system always selects the path through Router 2, ignoring delay variations and without any learning mechanism

The adaptive approach allows the Smart Router to respond to network changes, while the naive approach assumes static conditions and does not react to variations in delay.

C. Results Analysis

1) Observed Delay Trends: Figure 2 demonstrates the delay fluctuations during the adaptive and naive experiment. Router 2 initially has the lowest delay, but its delay spikes between iterations 20-40 due to the overhead. The Smart Router detects

this and reroutes traffic accordingly for the addaptive experiment. Also, when delay increases in the path, we can see that the path selection probability drops between iterations 60 and 70, but it does not surpass Router 2.

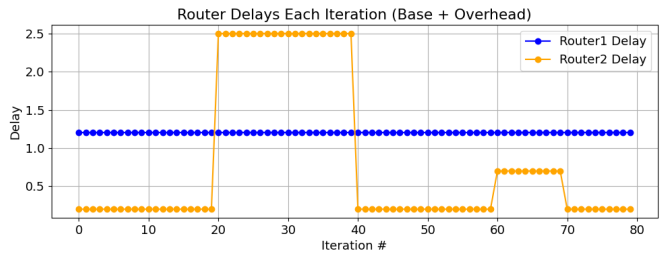

Fig. 2. Delay evolution in the adaptive routing approach

2) Routing Probability Evolution: Figure 3 shows how the routing probabilities evolve over time in the adaptive case. Initially, Router 2 is favored due to its lower base delay. However, when an delay is introduced at iteration 20, Router 1's probability increases as the system learns to avoid the higher latency. After iteration 40, when the overhead is removed, Router 2 gradually regains preference.

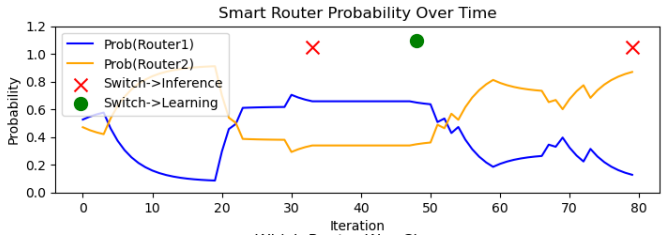

Fig. 3. Adaptive routing probabilities over time

In contrast, the naive approach maintains a fixed probability distribution, always favoring Router 2. This rigid strategy does not adapt to fluctuations in network conditions.

3) Reward: The reward shows that when path 2 has a higher probability, the reward also decreases. This trend holds over time, as seen in the probability graph. When the probability increases, the reward increases as well, and when the probability drops, the reward follows accordingly. This alignment is illustrated in Figure 4.

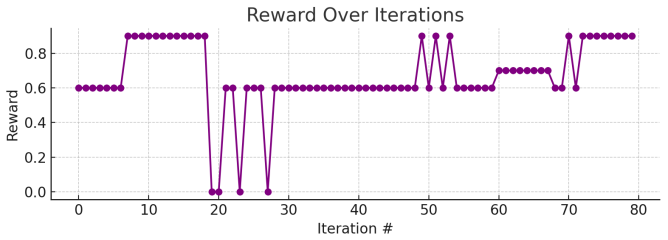

Fig. 4. Reward over time

4) Router Selection Over Time: Figure 5 illustrates the router selection decisions under the adaptive model. During the initial phase, Router 2 is primarily chosen. However, once additional delay is applied, the Smart Router shifts traffic towards Router 1. After iteration 40, Router 2 is once again selected more frequently as its delay returns to normal. However, a small probability margin is intentionally retained for Router 1 to allow occasional checks for potential changes in its status.

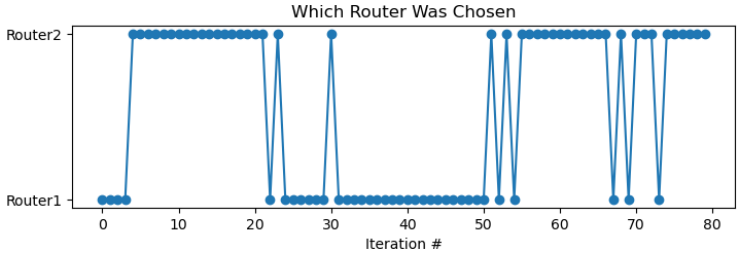

Fig. 5. Router selection dynamics in adaptive routing

Under the naive strategy, Router 2 is always selected, despite increased delay during iterations 20-40. This results in inefficient routing during periods of congestion.

5) Average Delay: As illustrated in Figure 6, the average end-to-end delay of the naive approach is significantly higher compared to the adaptive routing strategy. Specifically, the average delay using naive routing is 0.84 ms, whereas the reinforcement learning-based system achieves a lower average of 0.64 ms. This constitutes a reduction of approximately **31.25%** in delay when using the adaptive mechanism.

This improvement demonstrates that the Smart Router effectively learns to avoid paths with increased latency, even in dynamically changing environments. It is worth noting that this performance gain is achieved despite the limited duration of learning phases, highlighting the model's suitability for unreliable or time-varying networks.

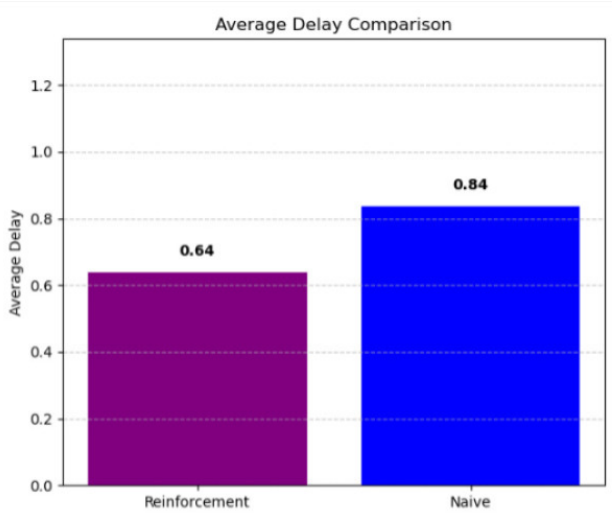

Fig. 6. Average delay per experiment

D. Discussion

The results indicate that the SLA-based adaptive approach effectively adjusts routing decisions based on observed delay, improving network responsiveness. The key observations include:

- **Adaptability:** The Smart Router shifts traffic away from congested paths, whereas the naive approach cannot.
- **Latency Reduction:** By dynamically updating routing probabilities, the adaptive model minimizes delays over time.
- **Phase Transitions:** The system effectively switches between learning and inference phases, allowing stability after convergence.

In contrast, the naive approach results in suboptimal routing, especially when unexpected delays occur. This highlights the benefits of reinforcement learning in adaptive network routing.

V. Conclusions and Future Work

This study demonstrates how an SLA-based reinforcement learning approach enables a Smart Router to dynamically adapt to changing network conditions. The system effectively learns to avoid high-latency paths and maintains optimal routing decisions through continuous feedback. The key findings include:

- **Adaptability:** The Smart Router dynamically adjusts routing probabilities in response to network changes, reducing latency.
- **Robustness:** The system transitions between learning and inference phases, ensuring stability once convergence is achieved.
- **Performance Gain:** Compared to a naive routing strategy, the adaptive approach reduces unnecessary delays by selecting the optimal router under varying conditions.

A. Future Work

While the current model effectively optimizes routing decisions based on delay, further studies could extend this framework by incorporating additional network metrics and expanding the simulation environment. Key areas for future research include:

- **Larger Network Topologies:** Expanding the system to support more complex network structures with multiple Smart Routers and routing paths.
- **Multi-Metric Optimization:** Integrating additional factors such as packet loss, bandwidth availability, and queue occupancy into the decision-making process.
- **Comparison with Alternative RL Strategies:** Evaluating the performance of different reinforcement learning models (e.g., Deep Q-Networks, Actor-Critic) for routing optimization.
- **Real-World Deployment:** Implementing and testing the adaptive routing mechanism on real network hardware or emulated environments such as Mininet, leveraging **P4** programmable switches and **Software-Defined Networking (SDN)** to enable flexible and adaptive forwarding decisions.

By addressing these aspects, future research can further enhance the adaptability and efficiency of reinforcement learning-based routing mechanisms in dynamic networking environments.

References

[1] A. Manzoor, M. Hussain, and S. Mehrban, "Performance analysis and route optimization: Redistribution between eigrp, ospf bgp routing protocols," *Computer Standards Interfaces*, vol. 68, p. 103391, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548919300996

[2] A. Qadeer, M. J. Lee, and D. Nobayashi, "P4-based in-network rl inference for efficient flow-level bandwidth allocation," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 1247–1252.

[3] L. C. de Almeida, W. R. D. da Silva, T. C. Tavares, R. Pasquini, C. Papagianni, and F. L. Verdi, "Desired — dynamic, enhanced, and smart ired: A p4-aqm with deep reinforcement learning and in-band network telemetry," *Computer Networks*, vol. 244, p. 110326, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624001580

[4] T. M. Tshilongamulenzhe, T. E. Mathonsi, D. P. D. Plessis, and M. Mphahlele, "Intelligent traffic routing algorithm for wireless sensor networks in agricultural environment," *Journal of Advances in Information Technology*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257143538

[5] H. Liu, W. Huang, X. Zhou, and X. H. Wang, "A comprehensive comparison of routing metrics for wireless mesh networks," in *2008 IEEE International Conference on Networking, Sensing and Control*, 2008, pp. 955–960.

[6] M. Avgeris, A. Leivadeas, and I. Lambadaris, "A reinforcement-learning self-healing approach for virtual network function placement," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–5.

[7] S. Pouryousef, L. Gao, and D. F. Towsley, "Robust path selection in software-defined wans using deep reinforcement learning," *ArXiv*, vol. abs/2212.11155, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254926637