Intelligent Network Routing using Reinforcement Learning

TeamID: CU_CP_Team_10181
Mentor: Palwinder Singh

Team Leader: Ketul Kasodiya
Team Member 1: Pranjal Panchal
Team Member 2: Sakshi Makwana

1. Abstract

This project presents an intelligent routing framework using Q-Learning, a reinforcement learning algorithm designed to optimize packet delivery in simulated computer networks. Traditional routing protocols such as Dijkstra's Shortest Path rely on static, precomputed paths and fail to adapt to dynamic traffic conditions or congestion. To overcome these limitations, the routing environment is modeled as a Markov Decision Process (MDP), where each router functions as a learning agent that selects next-hop decisions based on evolving Q-value estimates.

A medium-sized NSFNET-style topology was constructed using NetworkX, and a tabular Q-Learning agent was trained over multiple learning cycles with rewards for successful delivery and penalties for loops, drops, and latency. After training, the agent was evaluated against Shortest Path and Random routing under deterministic packet streams. Results show that Q-Learning achieves a consistently high Packet Delivery Ratio, matching the reliability of Shortest Path routing, while maintaining competitive hop counts and providing noticeably better balance in link usage.

Scaling experiments further confirmed that performance remains stable even as traffic load increases. Visualizations, including heatmaps and edge-usage overlays, highlight the agent's ability to distribute traffic intelligently across the network. Overall, this project demonstrates the strong potential of reinforcement learning for developing adaptive, efficient, and congestion-aware routing strategies suited for modern network environments.

2. Introduction

2.1 Motivation

Modern networks face unpredictable challenges: sudden traffic bursts, link failures, varying latency, and congestion. Classical routing algorithms (e.g., Dijkstra, Bellman-Ford) compute

deterministic least-cost paths but lack adaptability. With growing network scales and evolving demands, there is a strong need for routing mechanisms that:

- Learn from experience,
- Adapt to congestion,
- Avoid overloaded paths, and
- Make dynamic, decentralized decisions.

Reinforcement Learning (RL) provides a powerful framework for autonomous decision-making based on interactions with the environment, making it an ideal candidate for intelligent routing.

2.2 Problem Statement

Can a reinforcement-learning-based routing agent learn optimal next-hop decisions that match or outperform classical routing approaches under dynamic conditions?

The central challenge is whether an RL agent, without any preconfigured routing logic; can learn to deliver packets efficiently across a network.

2.3 Challenges

1. **State Representation:**

   The agent must make routing decisions only using (current_node, destination).

2. **Reward Design:**

   Rewards must encode successful delivery, loop avoidance, congestion penalties, and latency awareness.

3. **Exploration vs Exploitation:**

   The agent must explore new paths while converging to stable routes.

2.4 Objectives

1. Build a 14-node NSFNET-like simulation environment.
2. Implement tabular Q-learning for routing.
3. Train the agent through episodic learning.
4. Evaluate performance against Shortest Path and Random routing.
5. Perform scaling analysis by increasing packet generation rates.
6. Visualize link usage and behavior patterns through heatmaps and overlays.

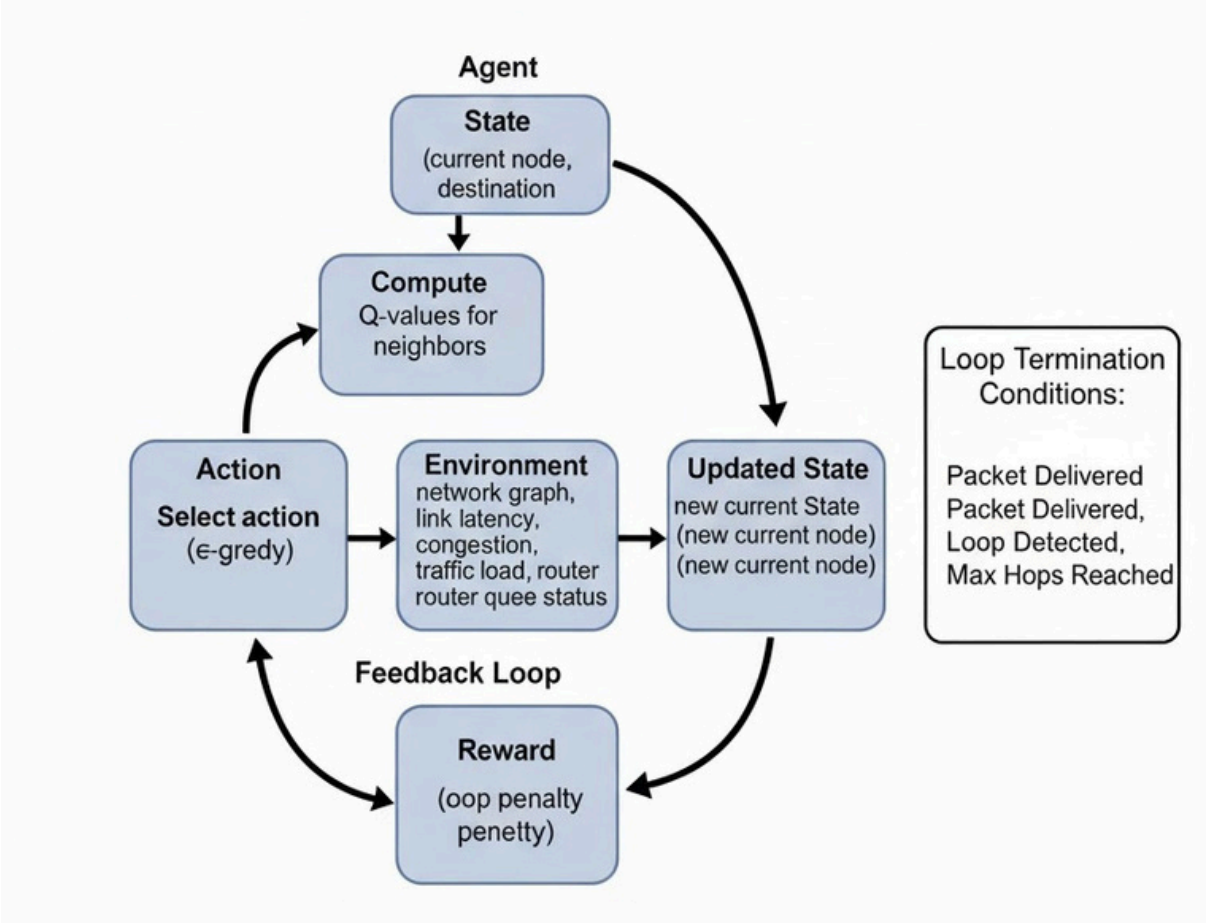
3. Methodology and System Architecture

3.1 Tools and Libraries

- **NetworkX** - graph modeling
- **NumPy** - numerical operations
- **Matplotlib** - visualizations
- **Pandas** - structured data representation
- **Tabulate** - tabular result formatting

All experiments were conducted in Python using Google Colab.

3.2 System Architecture



The core agent-environment loop is:

1. Agent receives state (current_node, destination)
2. Computes Q-values for neighbors
3. Selects an action using ε-greedy
4. Moves packet to next hop
5. Reward is computed

6. Q-table updated
7. Continue until delivered, looped, or max hops reached

3.3 Q-Learning Agent

State:

$$S = (current\_node, destination)$$

Action:

Any neighboring node of `current_node`.

Q-update rule:

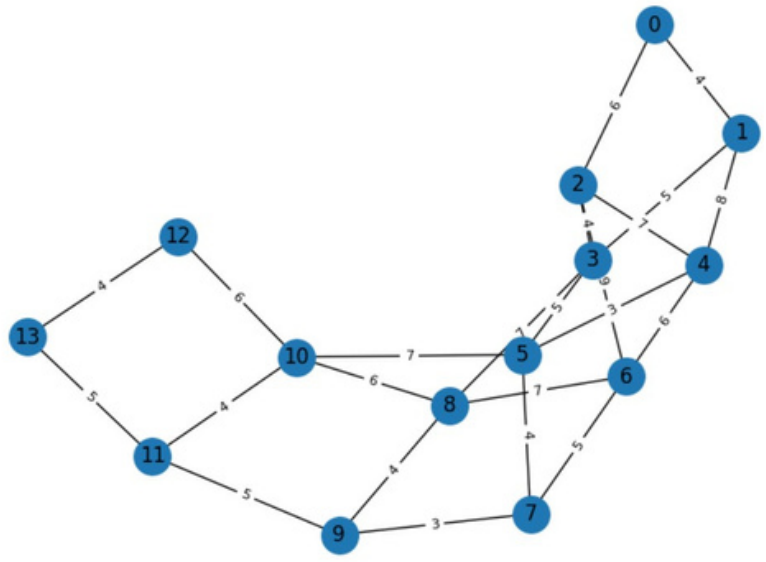
$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Hyperparameters

- α (Learning rate) = 0.6
- γ (Discount factor) = 0.9
- ε (Exploration) = 0.30 → decays to 0.05
- Episodes = 2000
- Max hops per packet = 20

4. Network Setup

4.1 NSFNET-like Topology



- 14 nodes
- 23 edges

- Weighted by latency (3–9 units)

- Built using NetworkX

This topology serves as the simulated backbone network.

4.2 Traffic Model

Adeterministic traffic generator produces packets:

- Across all 182 source–destination pairs
- Over 800 ticks
- With generation rate 0.02 (adjustable for scaling experiments)

4.3 Reward Model

Condition	**Reward**
Packet delivered	+20
Packet dropped	−20
Loop detected	−5
Hop latency penalty	−0.01 × latency

This encourages:

- Fast delivery
- Loop avoidance
- Low-latency path selection

5. Training and Evaluation

5.1 Training Behavior

Observations

- Values converge near +20, indicating frequent successful deliveries
- Negative troughs correspond to loops or dead ends
- ε-decay improves stability in later episodes

5.2 Evaluation: Q-Learning vs Shortest Pathvs Random

Agent	Generated	Delivered	Dropped	PDR	Avg_Hops	Mean_Link_Util
Q-learning (greedy)	2887	2887	0	1	2.629	0.20622
Shortest-Path	2887	2887	0	1	2.399	0.18818
Random	2887	1844	1043	0.639	8.21	0.97823

Results Summary

Agent	PDR	Avg Hops	Mean Link Utilization
Q-Learning	1.0	2.629	0.206
Shortest Path	1.0	2.399	0.188
Random	0.639	8.21	0.978

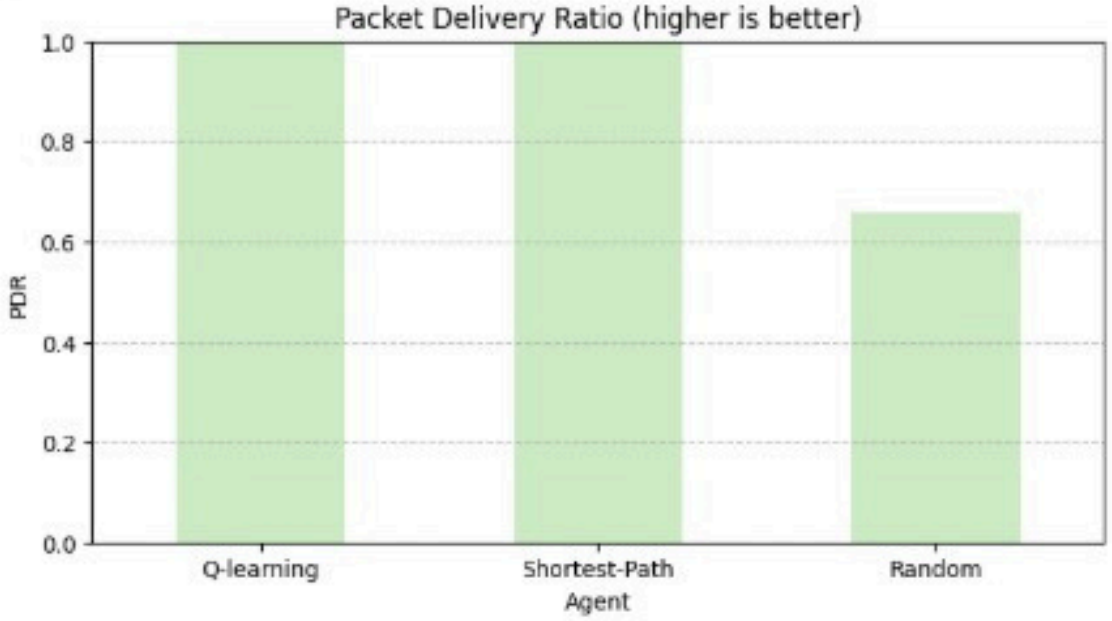
Interpretation

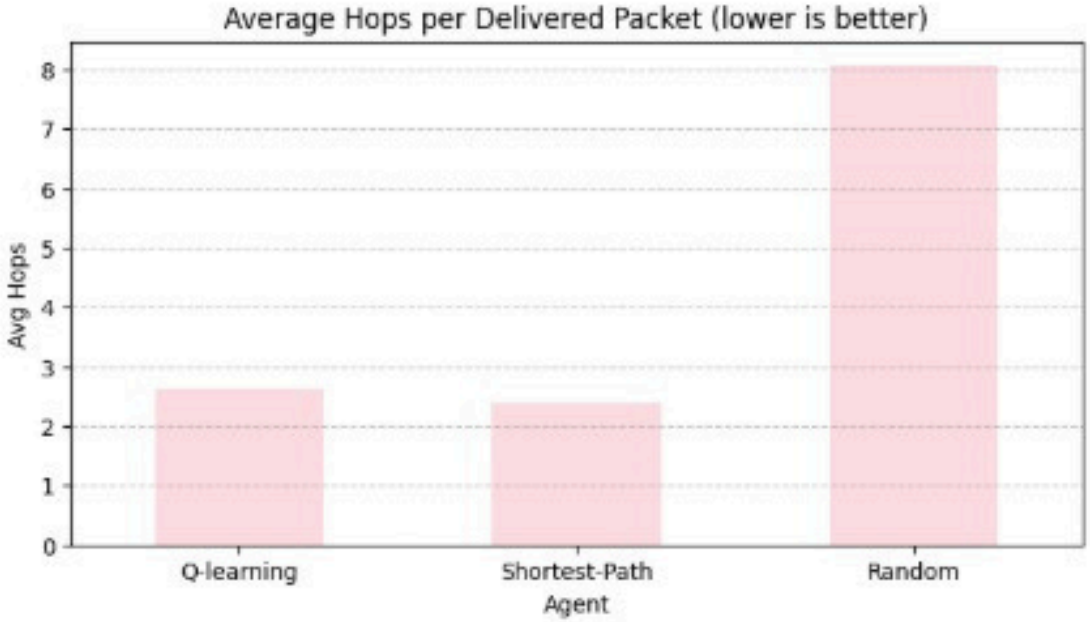
- Q-Learning achieves 100% PDR equal to Shortest Path
- Slightly higher hop count than SP but still optimal
- Much more balanced link usage
- Random performs poorly (high drops, high congestion)

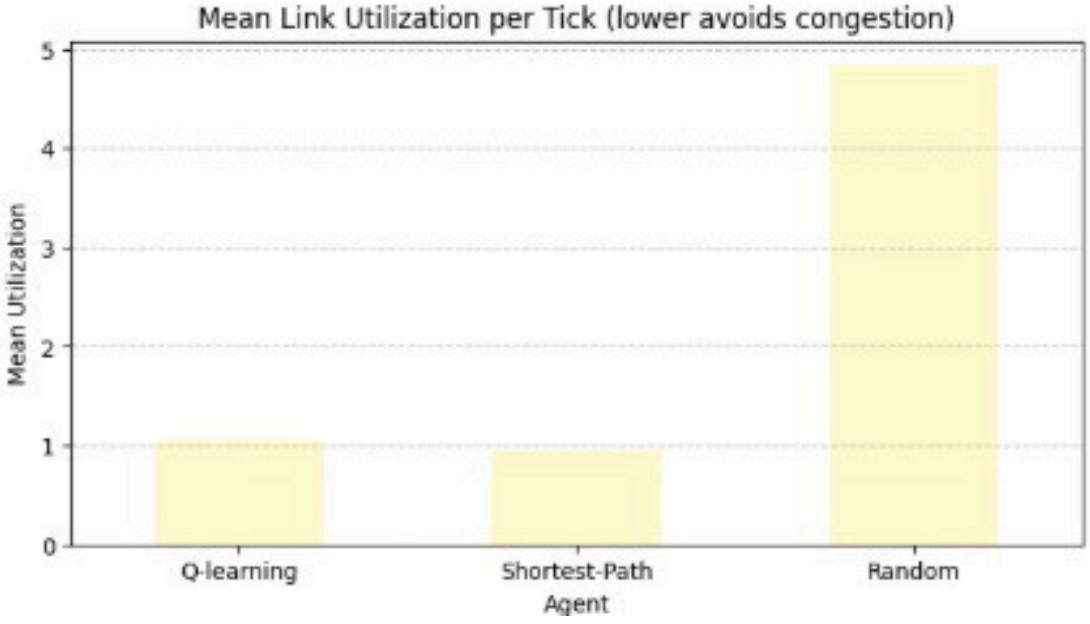
5.3 Metric Comparison (Visuals)

Key Insights

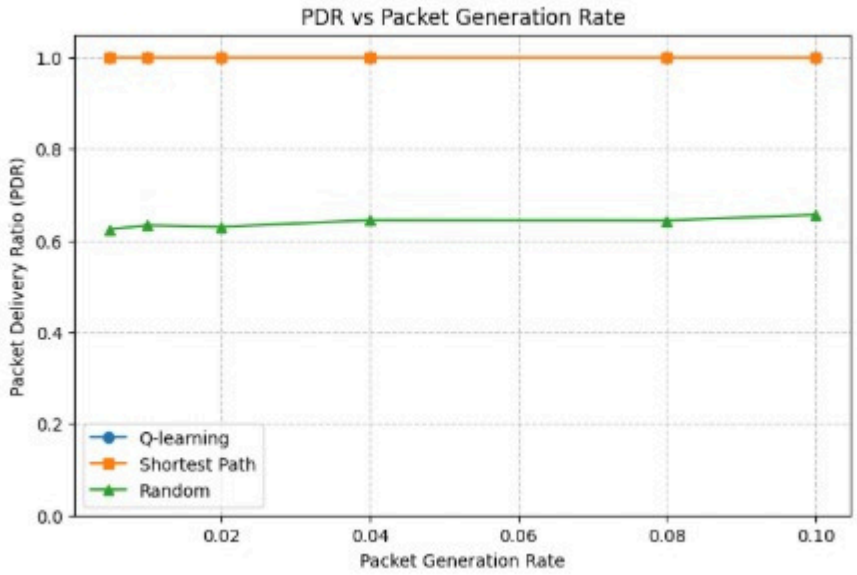
- Q-learning matches SP in delivery
- Chooses near-optimal hop counts
- Exhibits better load balancing
- Random routing overloads central links

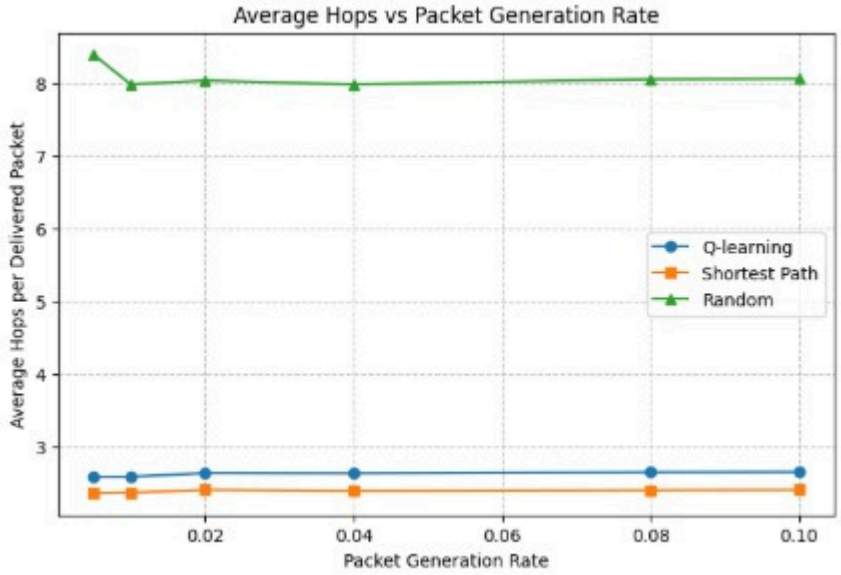

Packet Delivery Ratio (higher is better)
PDR
1.0
0.8
0.6
0.4
0.2
0.0
Q-learning
Shortest-Path
Random
Agent


Average Hops per Delivered Packet (lower is better)
Avg Hops
8
7
6
5
4
3
2
1
0
Q-learning
Shortest-Path
Random
Agent


Mean Link Utilization per Tick (lower avoids congestion)
Mean Utilization
5
4
3
2
1
0
Q-learning
Shortest-Path
Random
Agent

5.4 Scaling Experiments





Traffic generation rate varied from 0.005 → 0.10.

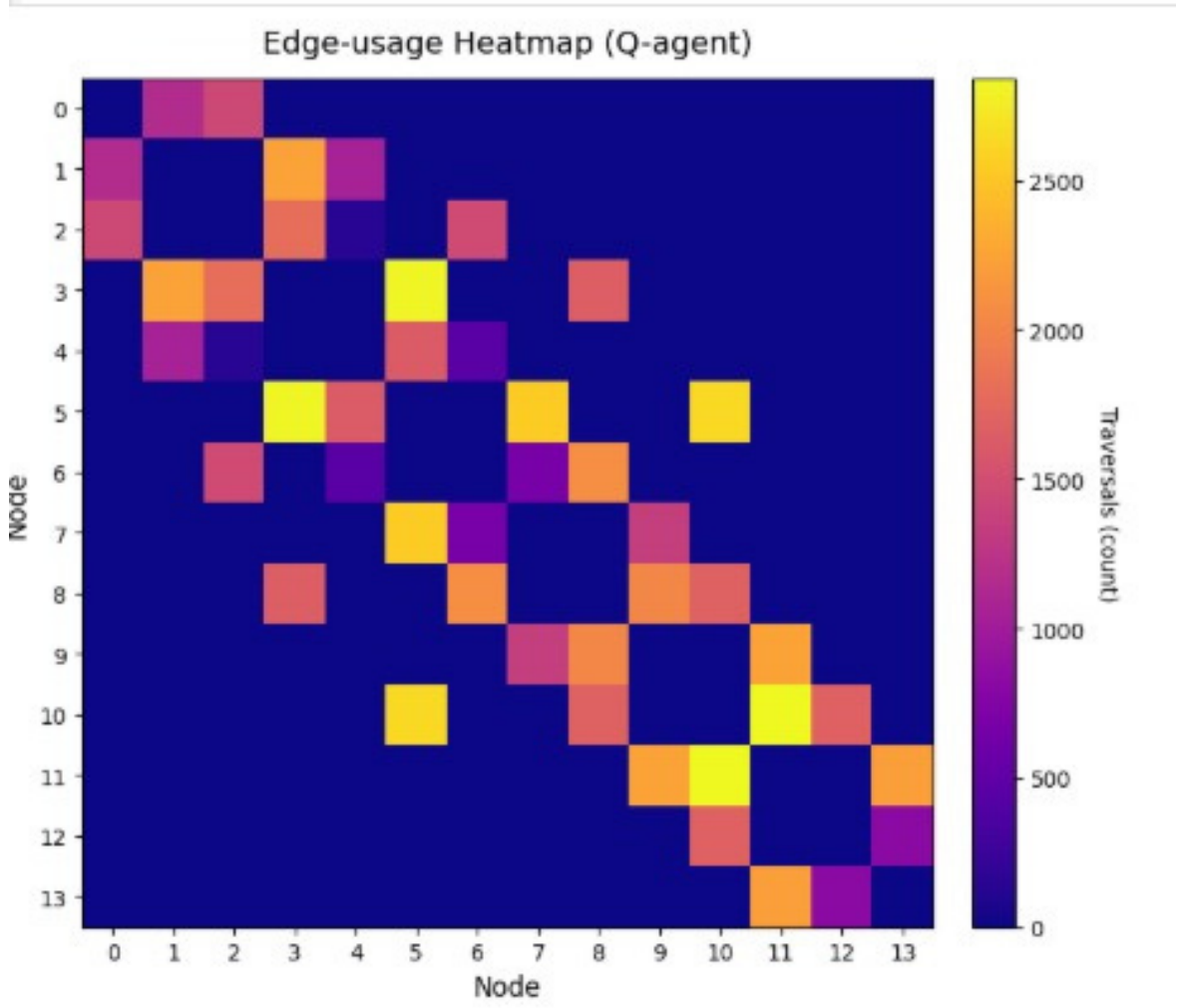
PDR vs Traffic Rate

- Q-learning maintains 1.0 PDR throughout
- SP also constant
- Random remains low (~0.63–0.65)

Avg Hops vs Traffic Rate

- Q-learning and SP stable
- Random route length fluctuates but remains high (~8 hops)

5.5 Link Usage Heatmap



Shows how often each link-pair is used.

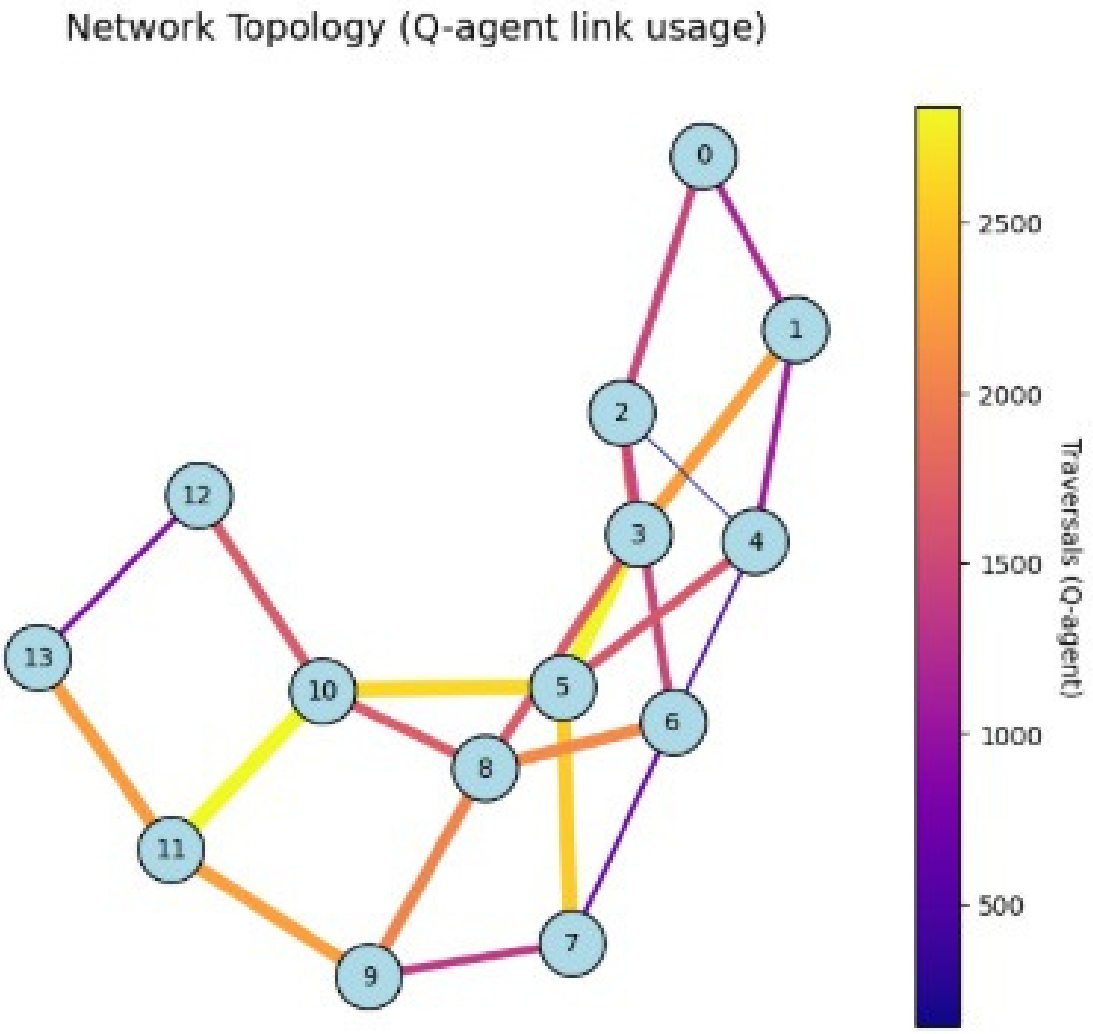
Insights

- RL agent avoids overused links
- Uses diagonally distributed traffic
- Shows awareness of congestion potential

5.6 Edge Usage Overlay

Highlights:

- Frequently used links in yellow/red
- Rarely used links in purple
- Q-agent distributes paths across the graph
- Avoids central bottlenecks (unlike SP, which always picks one path)



6. Results and Discussion

1. **Delivery Performance:**
   Q-learning achieved full delivery success across evaluations.

2. **Hop Count Efficiency:**
   Hop counts only slightly higher than the strictly optimal SP routing.

3. **Congestion Awareness:**
   Link utilization patterns show intelligent load distribution.

4. **Adaptability:**
   RL routing adapts to dynamic changes via experience, unlike SP.

5. **Learning Stability:**
   Converged Q-values indicate consistent policy development.

7. Conclusion

This project successfully demonstrates the viability of reinforcement learning for autonomous packet routing. The Q-learning agent:

- Learns efficient next-hop decisions independently
- Matches Shortest Path in delivery quality
- Shows improved traffic balancing
- Maintains performance under traffic scaling
- Avoids loops, congestion, and suboptimal routing choices

The results confirm that RL-based routing is a promising approach for future intelligent networks capable of adaptation and self-optimization.

8. Future Enhancements

1. **Deep Q-Learning (DQN):**
   Use neural networks to support larger topologies with continuous states.

2. **Integration with SDN Controllers:**
   Combine RL with centralized SDN control for real-time visibility and updates.

3. **Multi-Agent RL:**
   Allow all routers to learn collaboratively instead of one centralized agent.

4. **Predictive Congestion Control:**
   Use ML forecasting models to anticipate traffic surges and reroute proactively.