Graph Convolutional Reinforcement Learning for Collaborative Queuing Agents

Journée NetOpt 2022

Hassan Fawaz – Telecom SudParis

Julien Lesca, Pham Tran Anh Quang, Jérémie Leguay, Djamal Zeghlache, and Paolo Medagliani

November 10th, 2022

Contents

- ❑ Problematic
  - Traditional approaches are lagging behind
  - Need to add quality-of-service to queue management
- ❑ Related Works
  - On Machine Learning for Network Traffic Management
- ❑ Our approach
  - A DGN-aided Weighted Fair Queuing algorithm
  - DQN centralized and distributed benchmarks
  - Target networks and experience replay

Contents

Problematic

- There is a need for novel approaches to queueing and load balancing
  - Traditional techniques are failing to keep up with the demand
  - Higher throughput values and lower delays are expected
- Active queue management as a smart networking tool
  - Algorithms such as RED, CoDel, PIE, etc help reduce congestion and delay
  - They do not help with QoS or SLA guarantees
- Deep Reinforcement Learning for smart queue management
  - In our work we aim to use deep reinforcement learning to smartly manage queues
  - A DRL agent would decide how packets are being served
  - Its objective is to meet stringent requirements (throughput and delay) for a set of classified flows

Related Works: On RL- based traffic management

Discipline	Principle	Comments
[1]	• DQN learns when to drop/serve packets	• Reduces delay and jitter • High packet drop rate
[2]	• DQN to select optimal paths in MPTCP	• Better utilization of network • Increases aggregated throughput
[3]	• RL delay-based fairness scheduler	• Maximizes QoS efficiency
[4]	• RL decides what instance of the request traffic needs to be processed	• Deals with bursty traffic • Improves average wait time
[5]	• DDPG algorithm is used to distribute packets over multiple paths	• Actor-critic framework • Addresses scheduling of MPTCP
[6]	• Deep RL to dynamically learn the optimal buffer size	• Smaller queues w.r.t AQMs • No loss in throughput

* References cited on the final slide

SD-WAN Use Case

- An enterprise network headquarters (HQ), and five remote branches are interconnected
- By MPLS and broadband internet links
- A controller is placed at the headquarter site
- Access routers (ARs) are responsible for the interconnection.
- The ML agents are placed on the numbered nodes
- The training is done centrally, and the execution is distributed in the aforementioned nodes

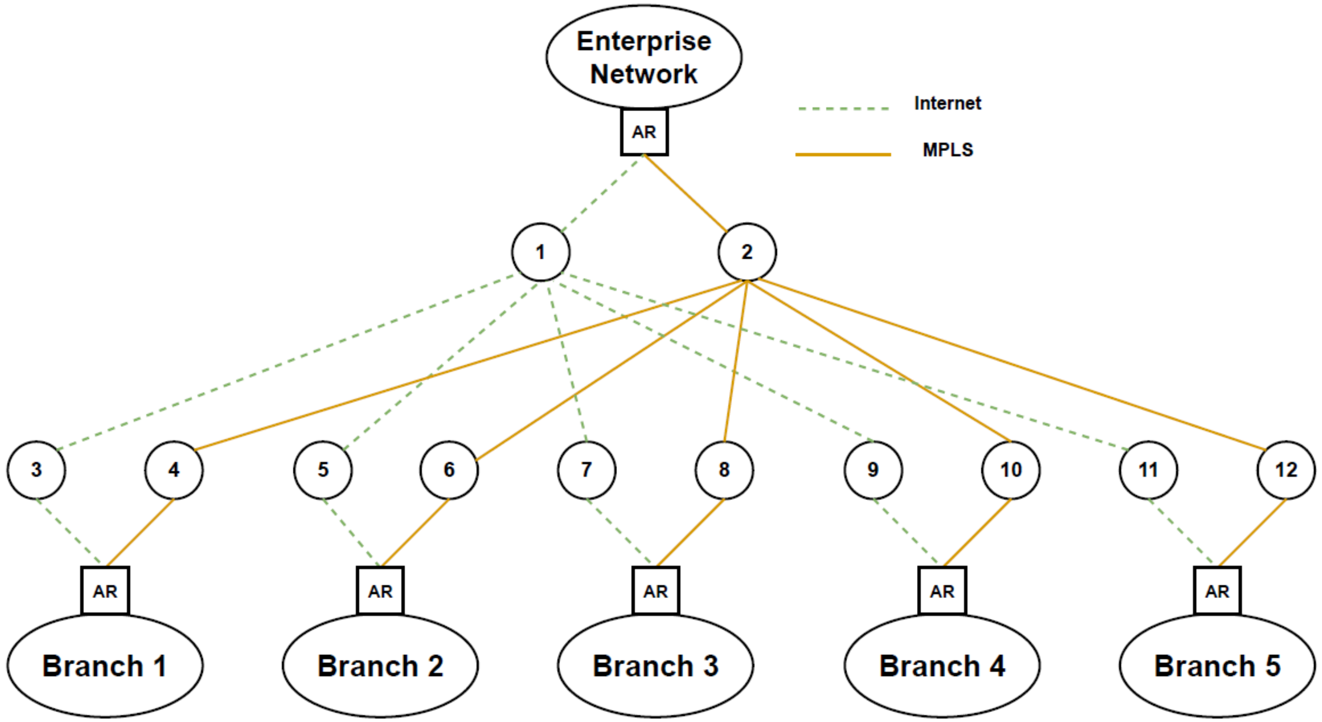


Figure 1. SD-WAN network with 5 branches

AR Device Structure

- The system architecture is split into two control entities
- In a slow loop, the routing agents takes decisions on how to balance the flows
- In a faster loop, a QoS agent applies a QoS policy, *i.e.*, the WFQ based RL approach we describe later
- In practice, we expect the routing control loop to be much slower than the QoS one
- This validates our current separation of the two tasks

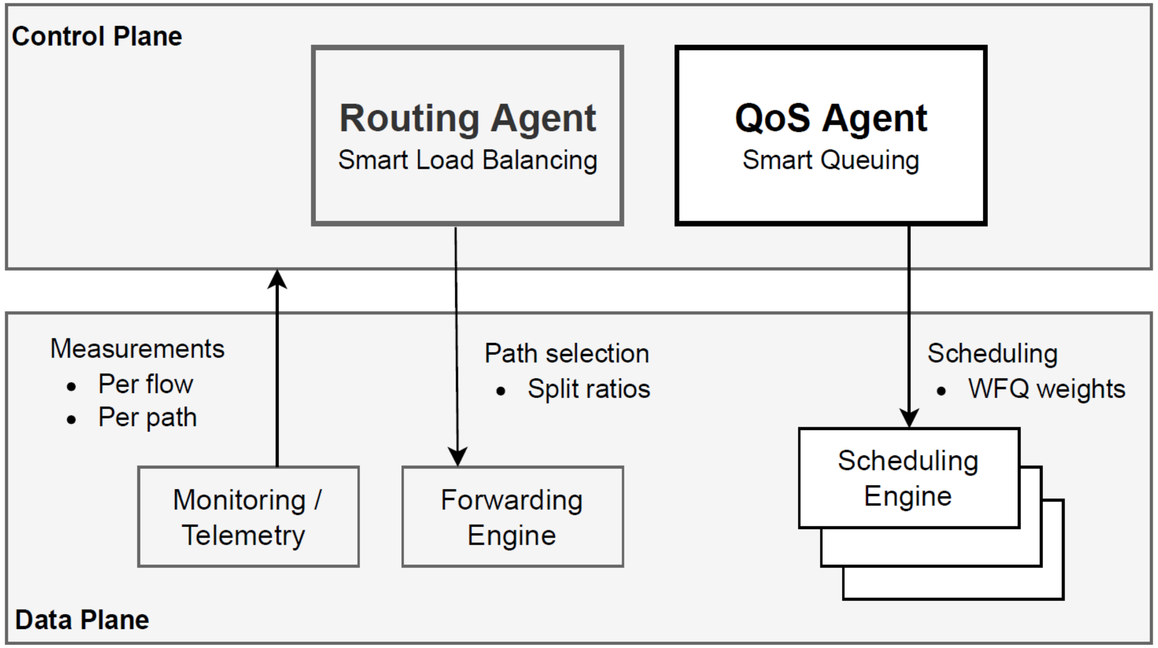


Figure 2. AR Device Structure

Our Approach: DRL assisted WFQ

- We implemented the WFQ approach illustrated in the figure below (in ns3)
- We classify network flows into three main groups: Gold, Silver, and Bronze in descending priority
- Gold, silver, and bronze groups for multimedia, business critical, and non-critical applications, respectively.
- We use a DRL agent, specifically a Deep Q-Network agent, to help optimize the weight selection continuously

Weight of flow *k*

$$R_k = \frac{w_k}{\sum_{i=1}^{K} w_i} R$$

Average rate of flow *k*

Total link capacity

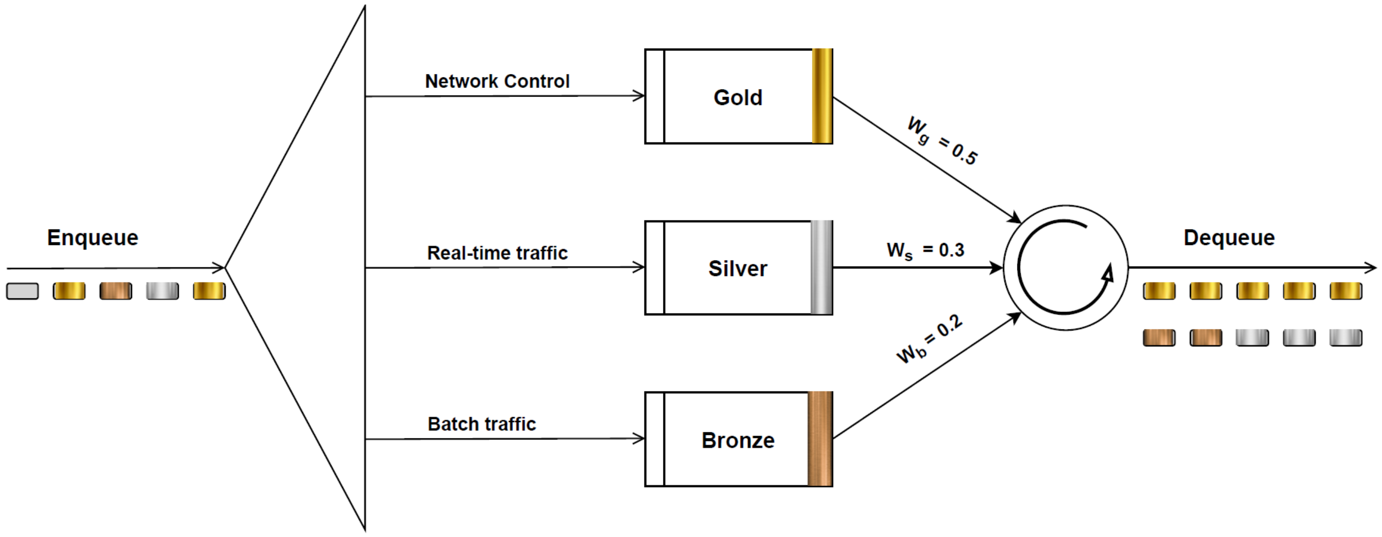


Figure 3. WFQ implementation

Graph Convolutional Reinforcement Learning-DGN

- Agents are situated at ingress nodes across the network, such as the numbered ones in Figure 1
- DGN combines the ideas of graph neural networks and deep reinforcement learning
- The agents are embedded in a graph G = (V,E), whose topology is related to the computer network in our scenario
- The existence of an edge between two agents in this graph means that they can exchange information
- Each agent has a set of neighbours, specified in what we call an adjacency matrix

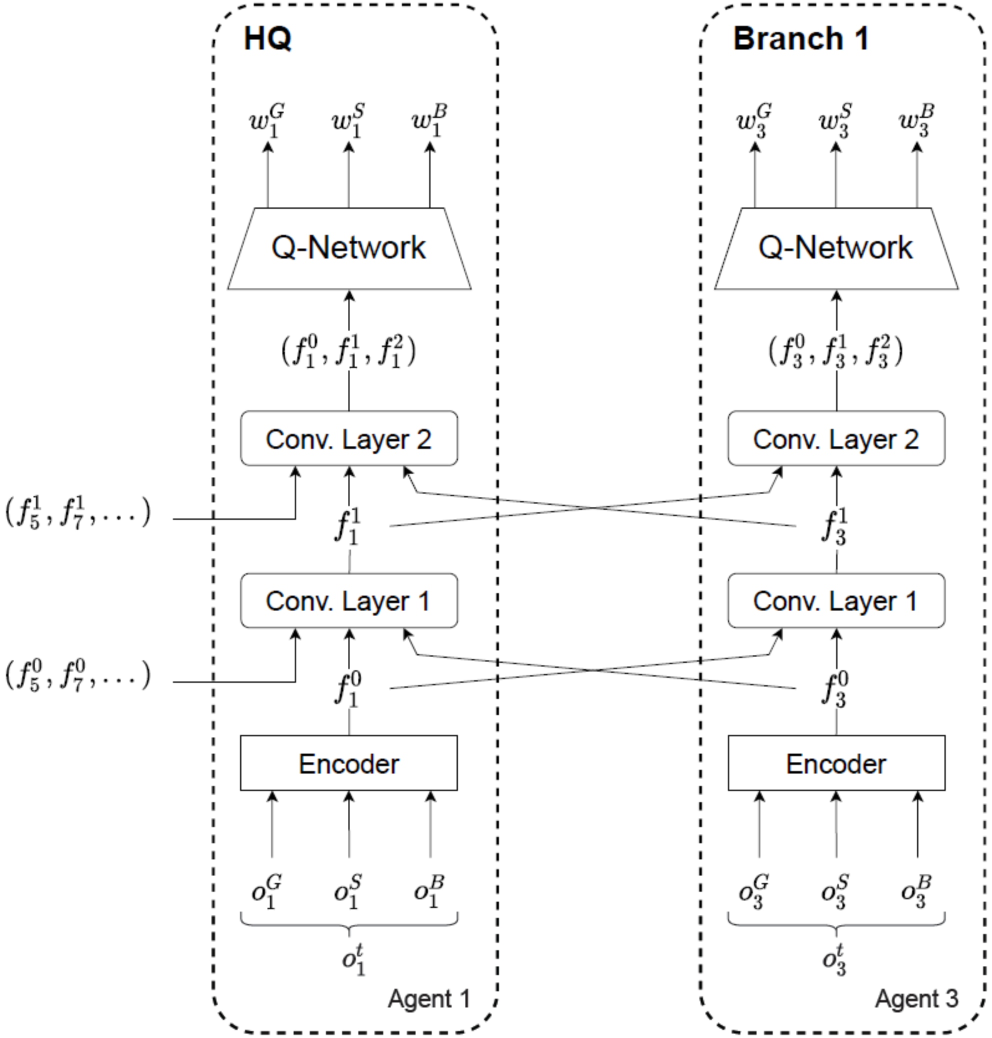


Figure 4. DGN Agents: HQ1 and Branch 1

Graph Convolutional Reinforcement Learning-DGN (2)

- DGN has three modules
- An encoder: A Multi-layer perceptron that takes the local observation and extracts relevant features
- A set of convolutional layers: which use attention mechanisms and helps learn how to best abstract relationships between agents
- A Q-network: The module which outputs the best actions on the weights with the objective of maximizing the long-term reward
- A multiple convolutional layer (*h*) module can be used
- The exchange of features between agents will permit the agents to obtain local knowledge from agents that are at a distance *h* from them

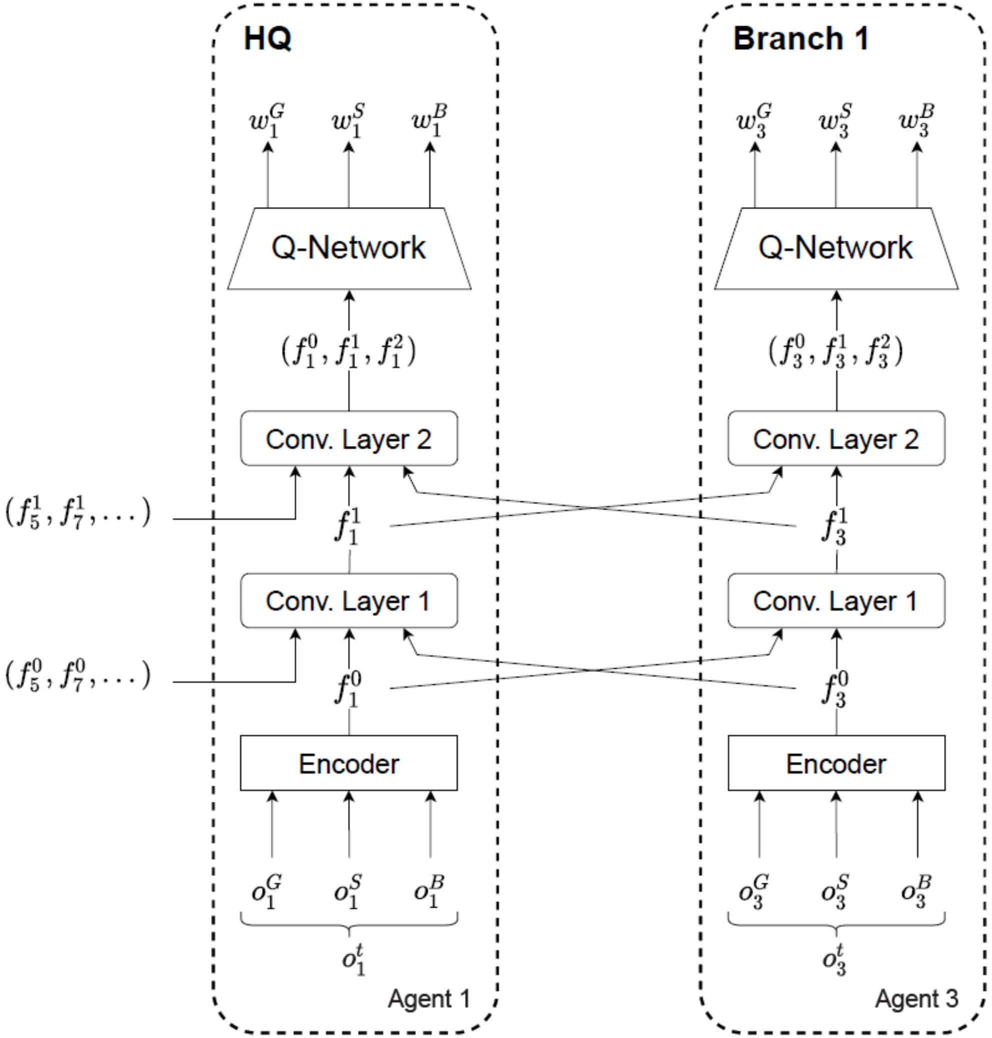


Figure 4. DGN Agents: HQ1 and Branch 1

Target Network and Experience Replay

DGN uses two classic deep learning mechanisms

❑ Target Network

- To avoid instability in training, we utilize a target network in addition to the main neural network
- This target is a copy of the main one and stores the Q-values
- In DGN we use a slow update

Target Network Parameters

$$\theta' = \tau\theta + (1 - \tau)\theta'$$

Main Network Parameters

Update Smoothness

❑ Experience Replay

- DGN implements a replay (experience) buffer, i.e., samples are stored in a memory and afterwards randomly sampled for training

Set of Observations

$$(\mathcal{O},\mathcal{A},\mathcal{O}',\mathcal{R},\mathcal{C})$$

Adjacency Matrices

Agent Actions

Next Set of Observations

Agent Rewards

Target Network and Experience Replay (2)

- With enough experiences in the buffer the training phase can begin
- We sample batches from the buffer and train with the objective of minimizing the loss

Loss

$$\mathcal{L}(\theta) = \frac{1}{S} \sum_{S} \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(O_{i,C}, a_i; \theta))^2$$

Batch Size

Q-value

Number of agents

$$y_i = r_i + \gamma \max_{a'} Q(O'_{i,C}, a'_i; \theta')$$

Discount factor

Description of the learning environment

❑ The observation

- The observation is represented by a tuple of six values: the throughput and delay values of each class of flows as served by the agent
- Because this is a continuous space, we discretize the values

❑ Possible actions

- At each step, the DGN agent acts on the weight of each class and either increases or decreases it
- With 3 flow groups considered. A total of 8 actions are possible
- The value of the increase/decrease is constant and pre-set as a parameter

Description of the learning environment (2)

❑ Agent Rewards

- The agent is rewarded every time it meets a requirement (throughput or delay) for any flow class
- It is penalized by the same value if it does not meet said requirement
- The reward could as such be in the negative, *i.e.*, a penalty
- Gold flows have higher reward values than the silver and the bronze, respectively

$$\omega_g^{th} \cdot \eta_g + \omega_g^{d} \cdot \phi_g + \omega_s^{th} \cdot \eta_s + \omega_s^{d} \cdot \phi_s + \omega_b^{th} \cdot \eta_b + \omega_b^{d} \cdot \phi_b$$

Binary values for meeting (+1) or violating (-1) the requirement

Reward values, constant, weighted by flow importance

Deep Q-Learning Benchmark: Distributed Approach

- Same set of states, observations, actions, and rewards as in DGN
- Does not embed attention or any innate agent cooperation mechanisms
- No inter-agent communications
- Incorporates target networks and experience replay buffers
- Trained by minimizing the loss

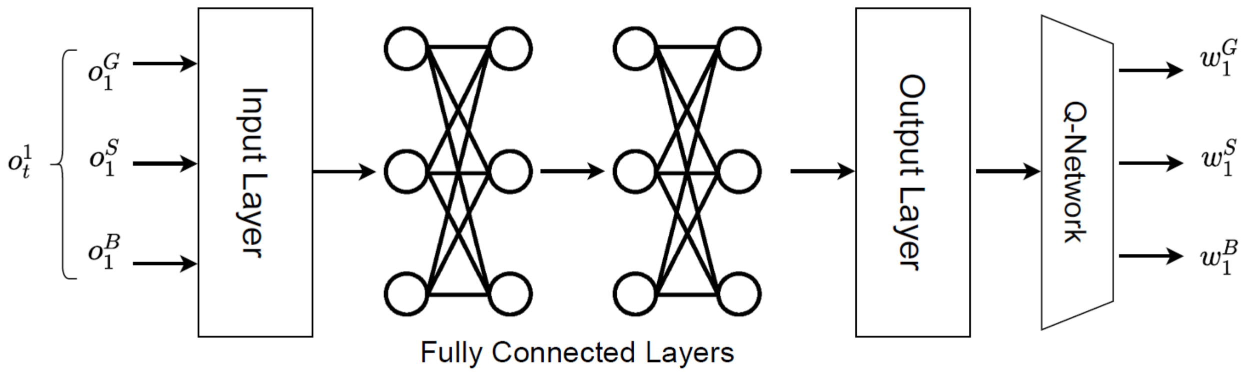


Figure 5. DQN agent

Deep Q-Learning Benchmark: Centralized Approach

- Same set of states, observations, actions, and rewards as in DGN
- Agent act as one unit
- They share the same set of states and observations
- They act on the environment collectively, and are issued a joint reward

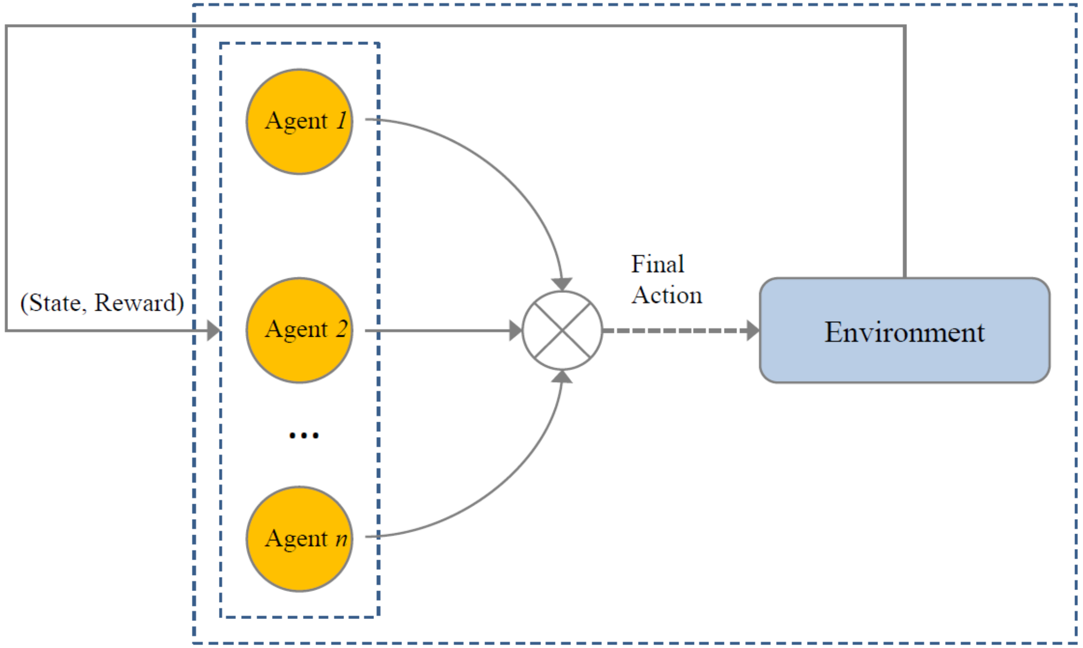


Figure 6. Centralized MADQN as proposed in [7]

Simulations and Results: Topology

- We consider that the load balancing is already performed
- The MPLS and internet sections are thus considered independtly
- We consider both UDP and TCP traffic
- The traffic sources are heterogenous to motivate agent cooperation
- Branch 1 produces gold and silver flows, Branch 2 silver and bronze, and so on
- The algorithms are simulated in an ns-3 enviroment

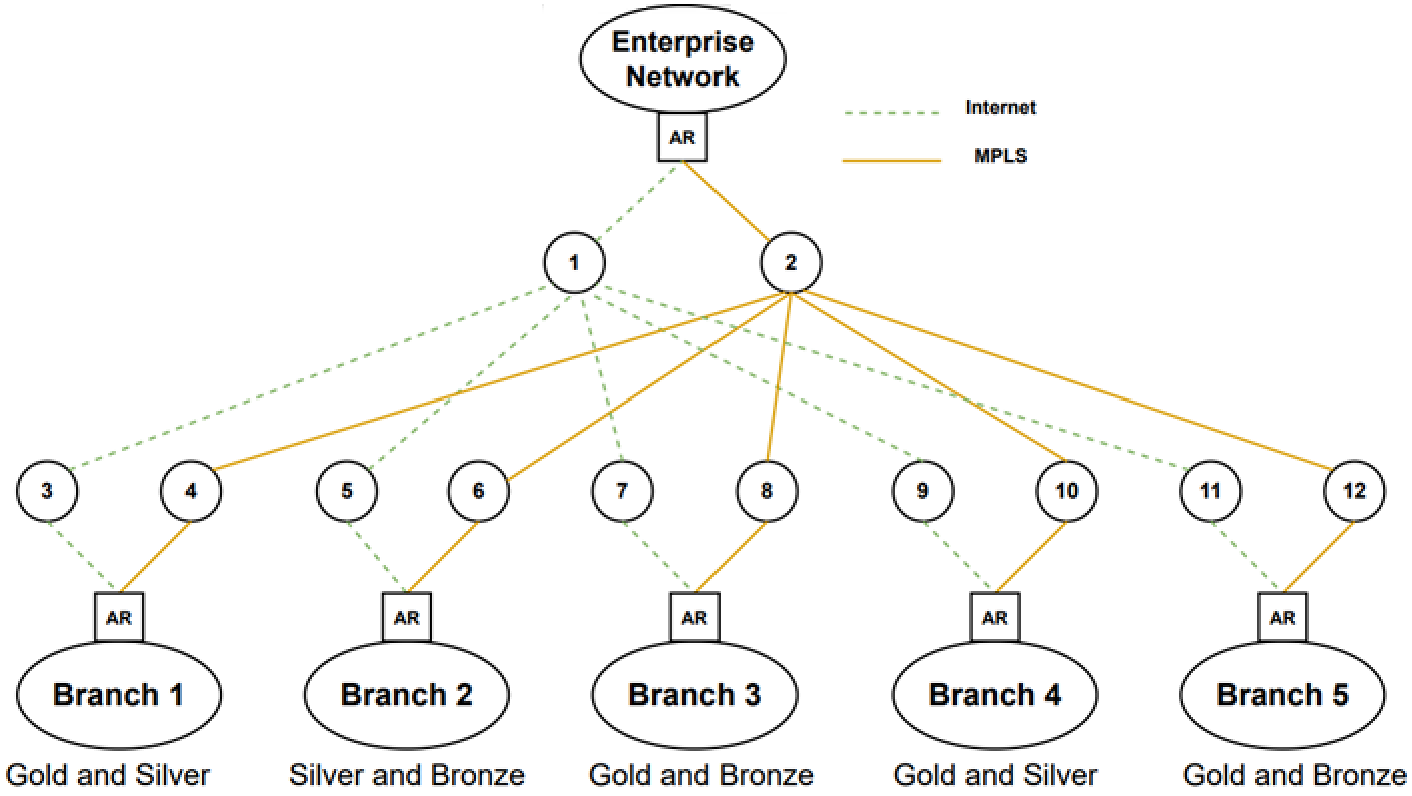


Figure 7. Considered topology for agent simulations

Simulations and Results: Benchmarks

Approach	Abbreviation	Agent Communication/Cooperation	Notes
Graph Convolutional Multi-Agent	DGN	Attention model, feature exchange	Small overhead/ signaling
Centralized Multi-Agent DQN	CDQN	Shared observations, actions, rewards	Extensive memory requirements
Decentralized Multi-Agent DQN	DDQN	No communication or cooperation	No overhead
Priority Queuing	PQ	No communication or cooperation	No overhead

- CDQN is unrealistic to implement and does not scale well at all
- DDQN is a benchmark considered to stress the importance of inter-agent cooperation

Simulations and Results: Topology Parameters

Table I: Parameters for the simulations

Parameter	**Value**
Number of O-D pairs	10, 4 gold, 3 silver, 3 bronze
Snapshot duration / # of snapshots	10 sec / 300
$T_g/T_s/T_b$	30 / 10 / 5 Mbps
$d_g/d_s/d_b$ for UDP	0.15 / 0.3 / 0.4 seconds
$d_g/d_s/d_b$ for TCP	0.1 / 0.15 / 0.2 seconds
Delay to throughput relevance $\kappa_g, \kappa_s, \kappa_b$	0.8
Reward relative to flows G/S/B	$3x/2x/x$
Rewards η, ϕ	300
WFQ weight update δ	0.03

Simulations and Results: Learning Parameters

Table II: Parameters for MADQN agents

Parameter	**Value**
Activation function	ReLu
N^o of fully connected layers	2 each with 128 neurons
Exploration rate ϵ	starts with 1 and decays to 0.001
ϵ - decay ϵ	multiplied by 0.99955 per episode
Discount factor γ	0.99
Training batch size	32

Table III: Parameters for DGN agents

Parameter	**Value**
N^o of neighbors	Depends on node position, up to 5
N^o of convolutional layers	2
Exploration rate ϵ	0.6 and not decayed for training
N^o of encoder MLP layers	2
N^o of encoder MLP units	(128,128)
Scaling factor τ	0.01
Discount factor γ	0.99
Training batch size	32

Simulation and Results: Agent Convergence

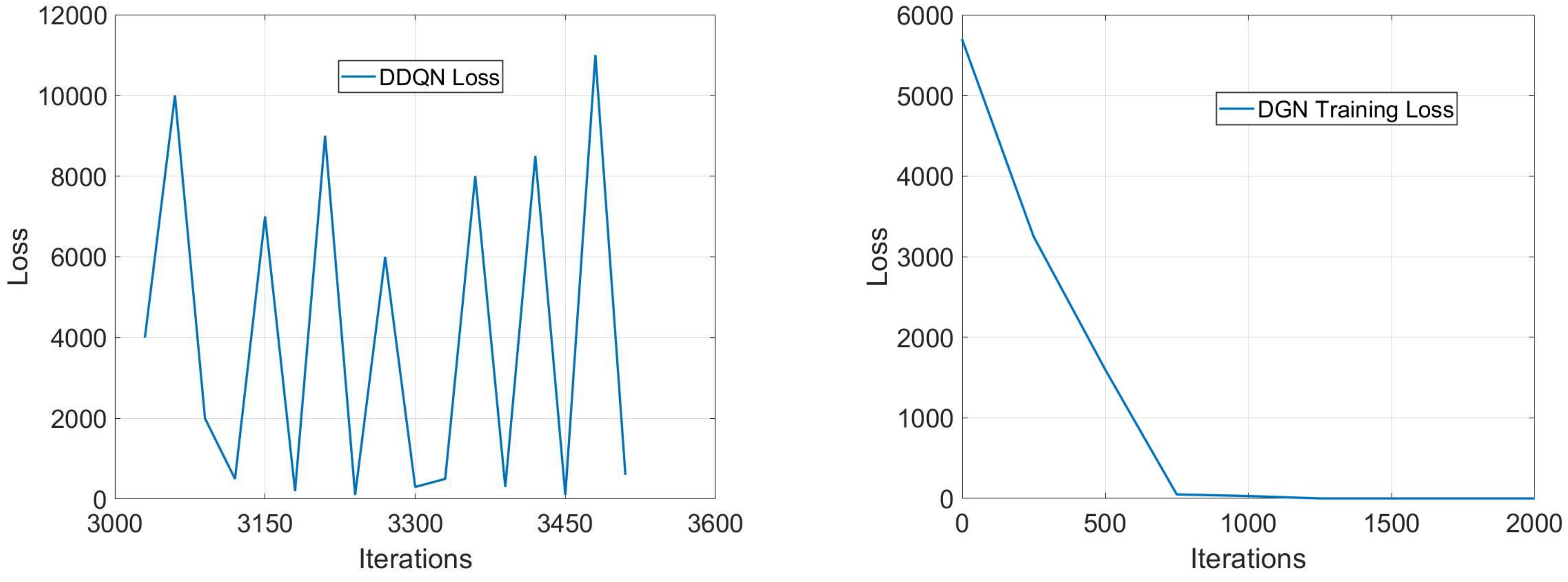
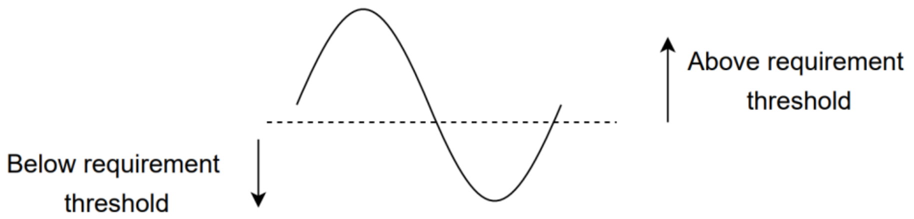


Figure 8. Agent convergence

- DGN convergence is verified by tracking the loss function
- Distributed DQN does not seem to converge even though the rewards tends to improve

Simulation and Results: UDP Traffic – Throughput (1)

- The throughput requirements are 5/10/30 Mbps for gold, silver, and bronze group flows
- DGN meets all these requirements
- Distributed MADQN does not with the bronze flows sitting at around 4 Mbps below the required mark
- The results reflect the lack of convergence in the case of DDQN



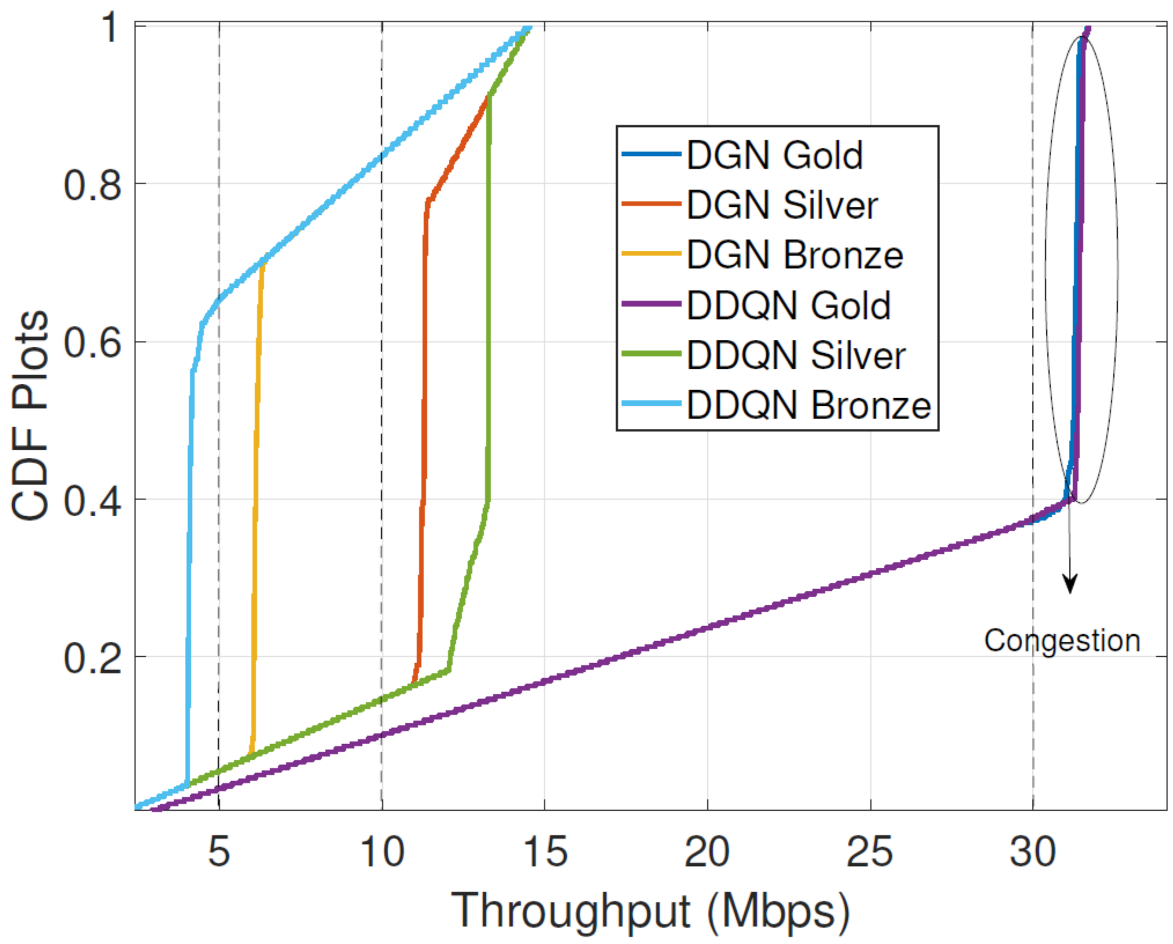


Figure 9. DGN vs DDQN, UDP, Throughput

Simulation and Results: UDP Traffic – Throughput (2)

- CDQN gives throughput results at around 5.5, 11.4, and 31 Mbps for bronze, silver, and gold flow groups
- CDQN meets all the required thresholds
- We compare to a classic Priority Queuing (PQ) algorithm
- PQ serves packet in the queue in strict descending priority
- PQ fails to meet the bronze flow group requirements which register a throughput at about 1 Mbps only
- The centralized nature of CDQN helps cover for the lack of agent communication observed with DDQN

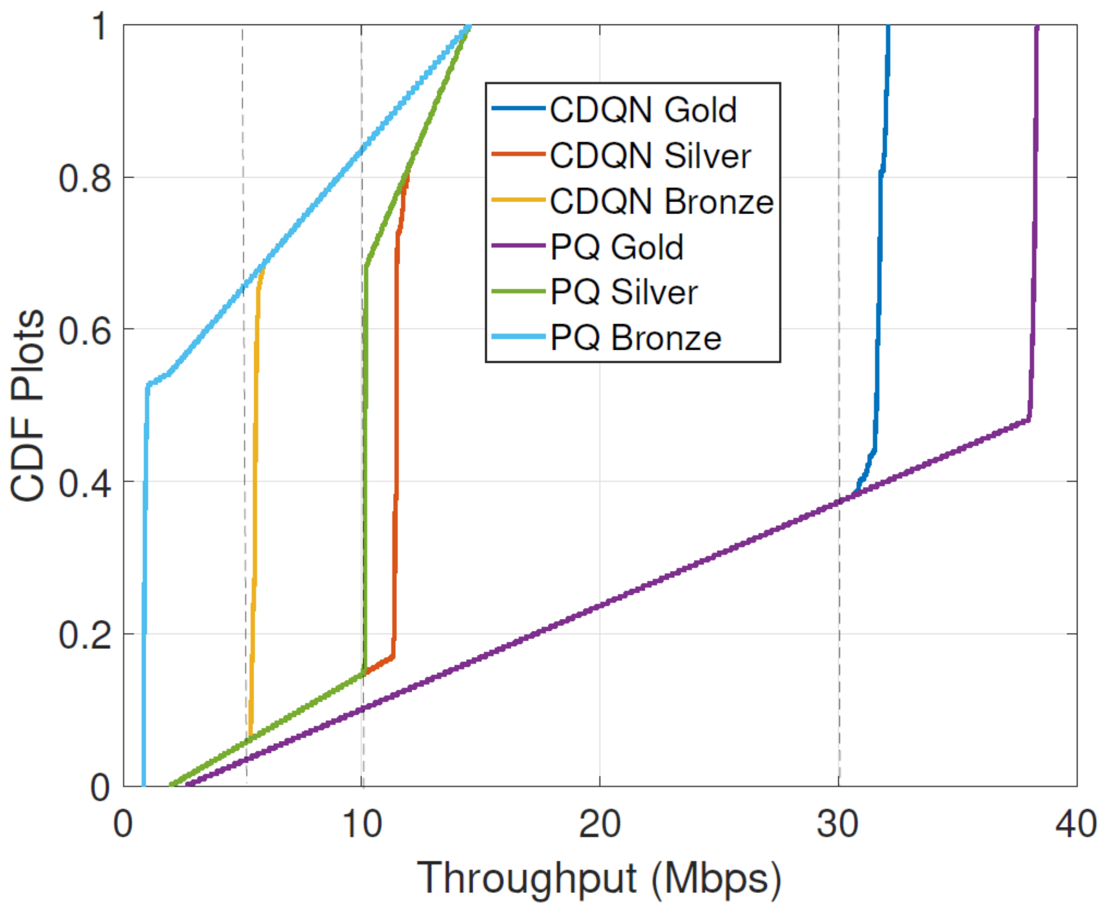


Figure 10. CDQN vs PQ, UDP, Throughput

Simulation and Results: UDP Traffic – Delay (1)

- The targets for the end-to-end delay for the flow groups are set at 0.15, 0.3, and 0.4 seconds for gold, silver, and bronze, respectively
- DGN was able to meet all these requirements
- For DDQN the silver flow requirements are violated in more than half the cases
- Out of 6 total constraints, DDQN meets half

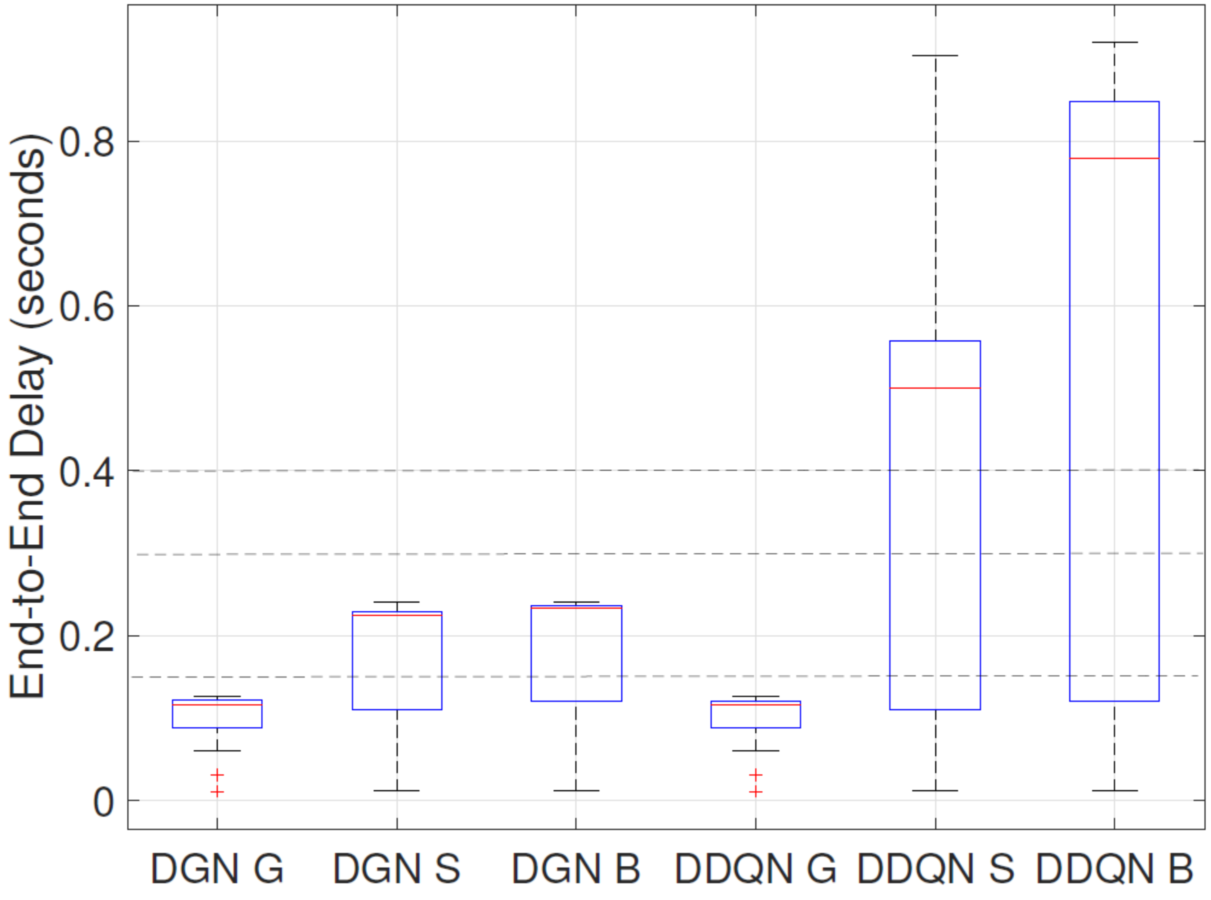


Figure 11. DGN vs DDQN, UDP, Delay

Simulation and Results: UDP Traffic – Delay (2)

- Similar to the throughput case, CDQN was able to meet the delay demands with median values at around 0.128, 0.209, and 0.298 seconds for the gold, silver, and bronze flow groups
- This is not the case for priority queuing where the maximum delay values are at around 2 and 3 seconds, respectively
- The centralized nature of this DQN implementation again helps meet the target demands
- PQ is not intelligent enough to meet the silver flow delay thresholds even though it can meet their throughput demands

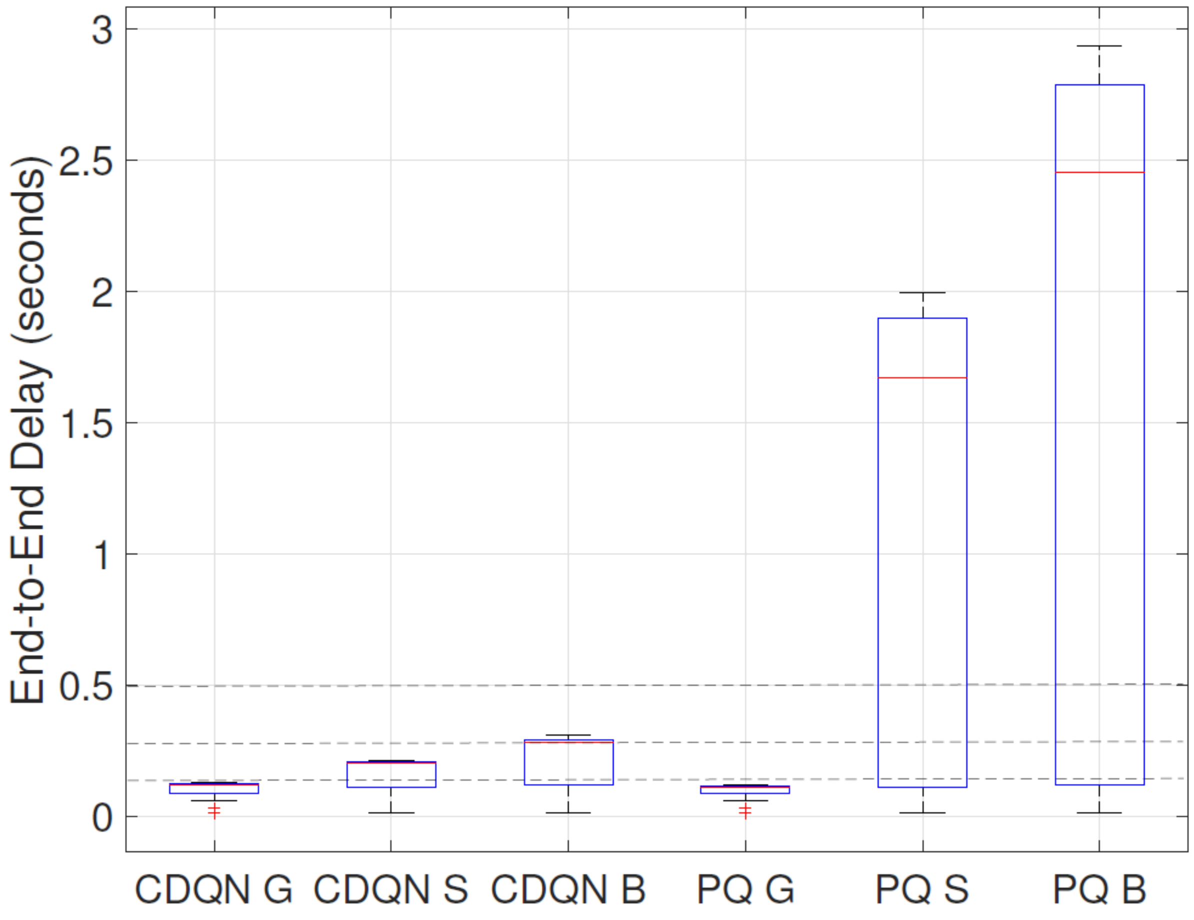


Figure 12. CDQN vs PQ, UDP, Delay

Simulation and Results: TCP Traffic – Throughput (1)

- We now consider TCP traffic
- The required throughput thresholds are still the same for all the flows at 5, 10, and 30 Mbps for the bronze, silver, and gold flows
- TCP traffic provides a more intricate scenario, the weights need to account for TCP's innate congestion control mechanism
- Nonetheless, DGN was able to meet the all the flow requirements, unlike DDQN which had the throughput values for the gold flows at about 28 Mbps

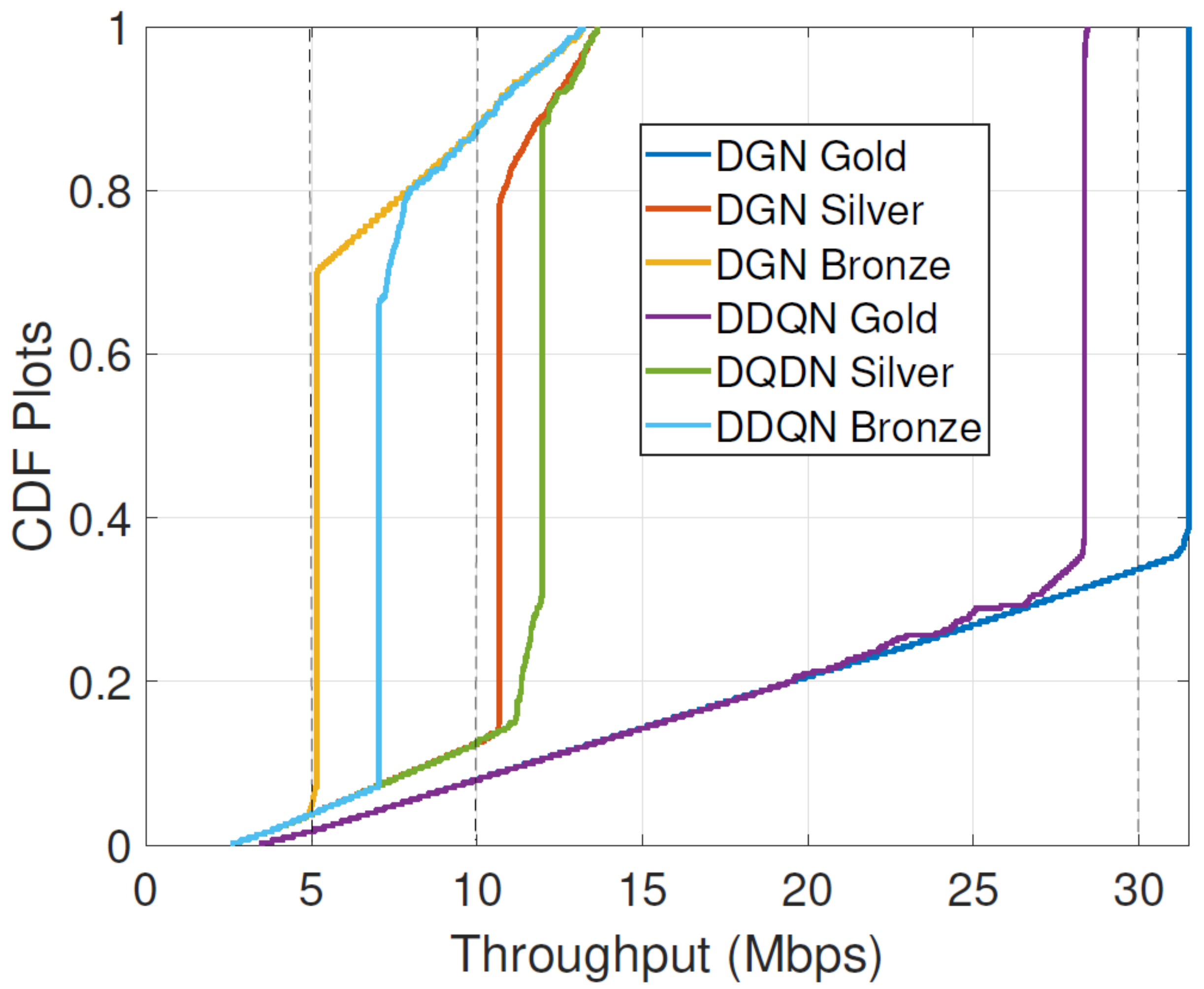


Figure 13. DGN vs DDQN, TCP, Throughput

Simulation and Results: TCP Traffic – Throughput (2)

- Again, CDQN can cover for the lack of agent communications problem present in DDQN
- The threshold for all the flows groups are met
- The same cannot be said for Priority Queuing which slightly violated the demands for the silver flows and significantly violated those of the bronze flows (at about 0.7 Mbps only)

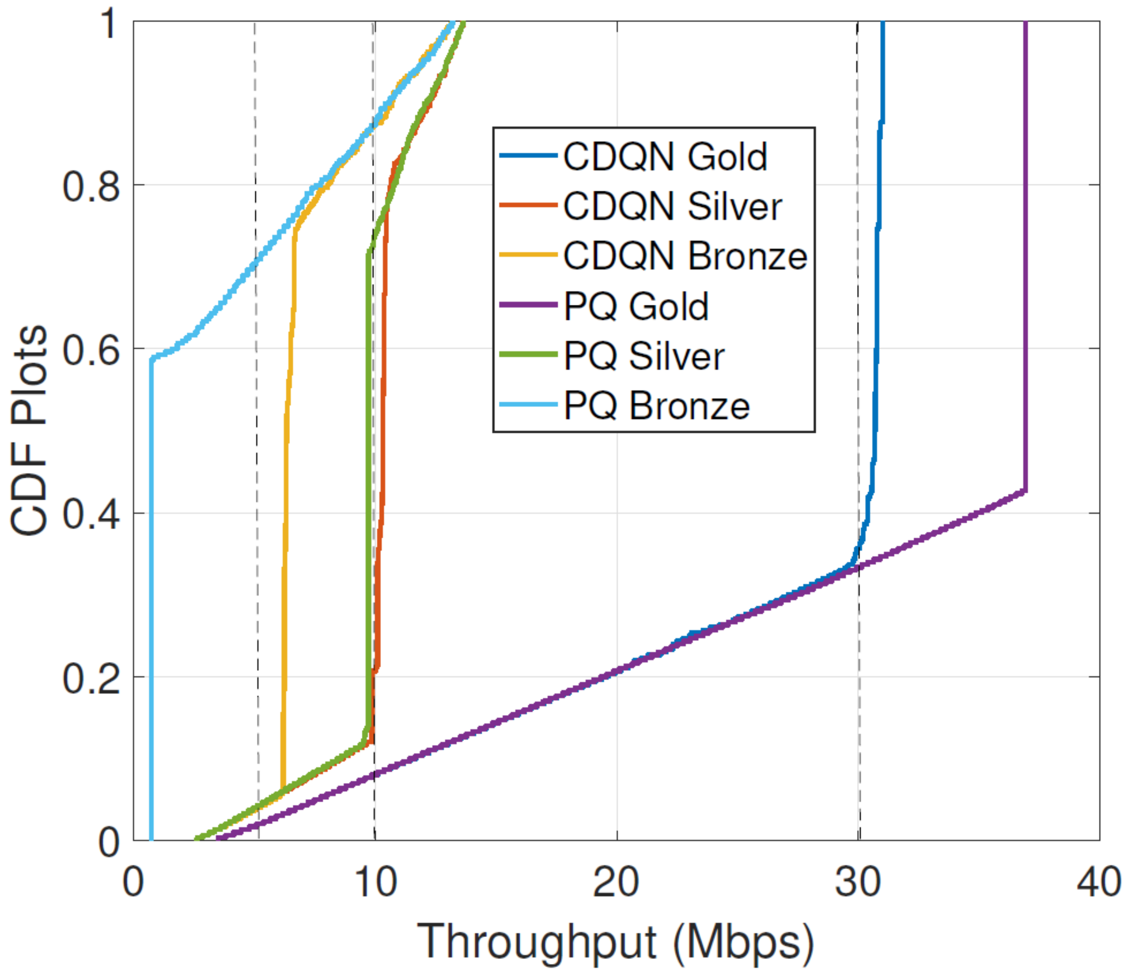


Figure 14. CDQN vs PQ, TCP, Throughput

Simulation and Results: TCP Traffic – Delay (1)

- The delay requirements are set at 0.1, 0.15, and 0.2 for the bronze, silver, and gold flows
- DGN is able to meet all with median values at about 0.04 seconds for the gold, 0.13 for the silver group flows, and slightly less than 0.15 seconds for the bronze flows
- DDQN is able in this case to meet the demands as well. Nonetheless it can be noted that the latter provides in general higher average delays while also exhibiting inconsistencies

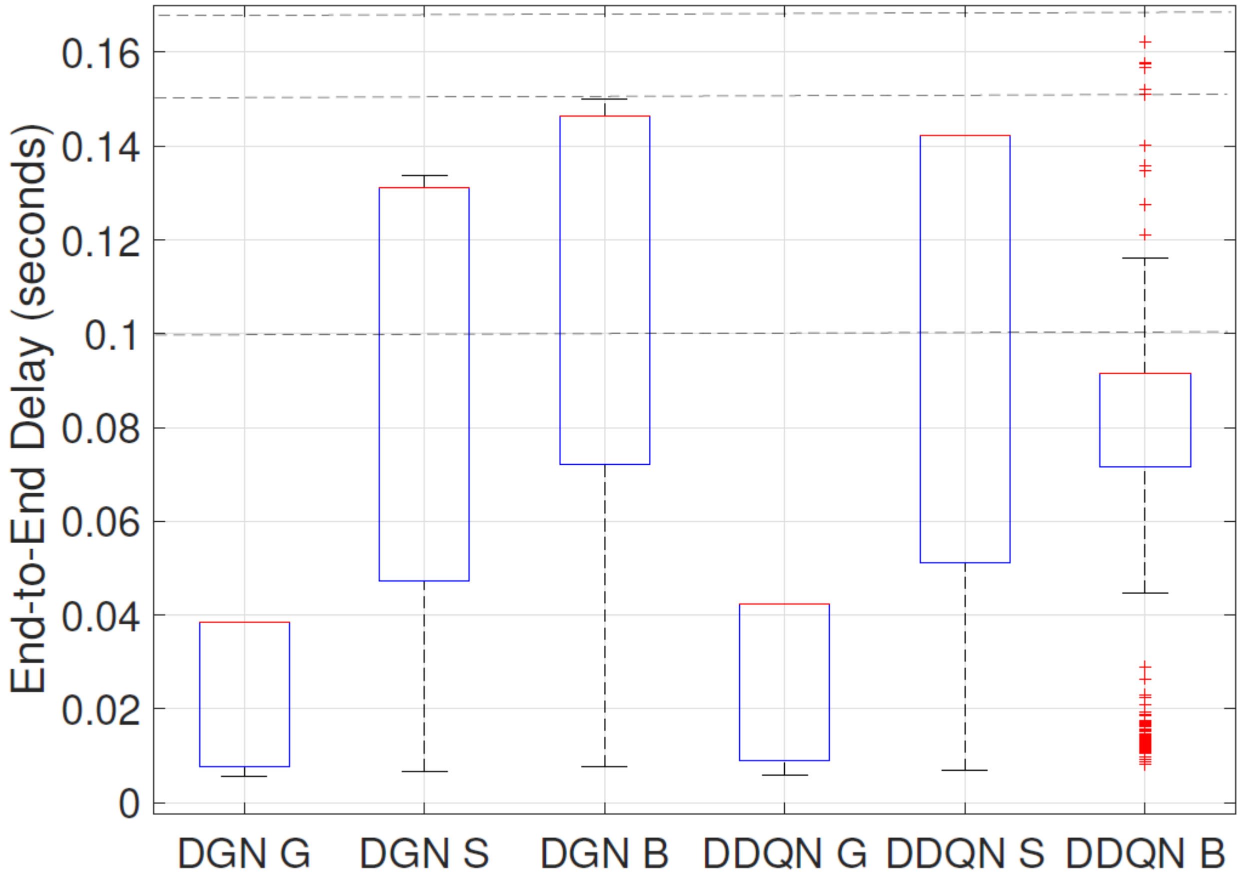


Figure 15. DGN vs DDQN, TCP, Delay

Simulation and Results: TCP Traffic – Delay (2)

- Finally, as in the case of UDP traffic, CMDQN is able to match DGN in terms of meeting the end-to-end delay requirements
- Priority queuing violates the requirements for both the silver and bronze flows in several instances

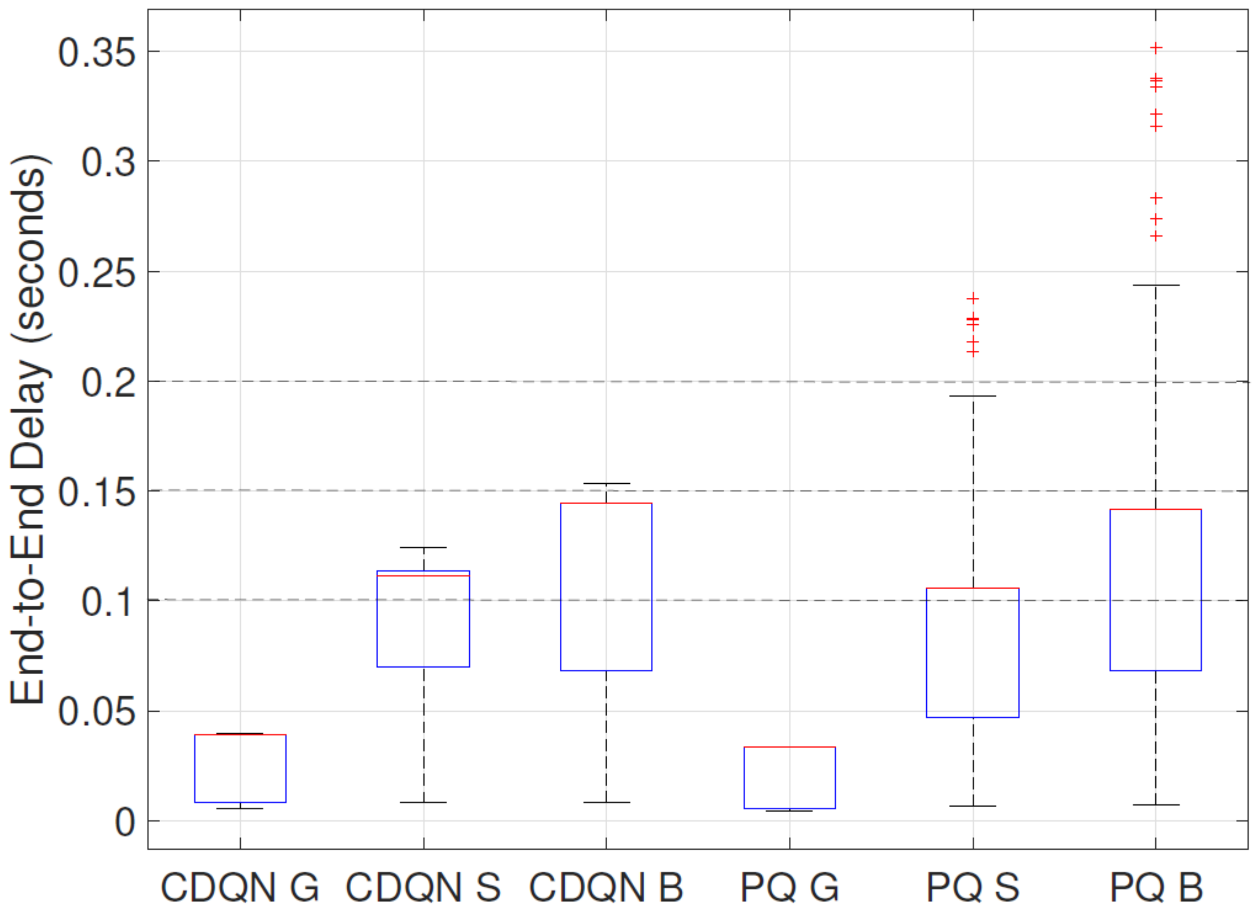


Figure 16. CDQN vs PQ, TCP, Delay

Simulation and Results: Scalability (1)

- We additionally seek to test our approach in a larger scale scenario
- To this end we consider a topology based on the ION-NY network topology seen in Fig. 17
- We consider 17 sources and 3 receiving nodes
- The agents are only present at peripheral nodes which are connected to hosts
- Inter-agent communications are possible with interconnected nodes and with the receiving end nodes as well
- The approach is this time build in Mininet, a network emulator

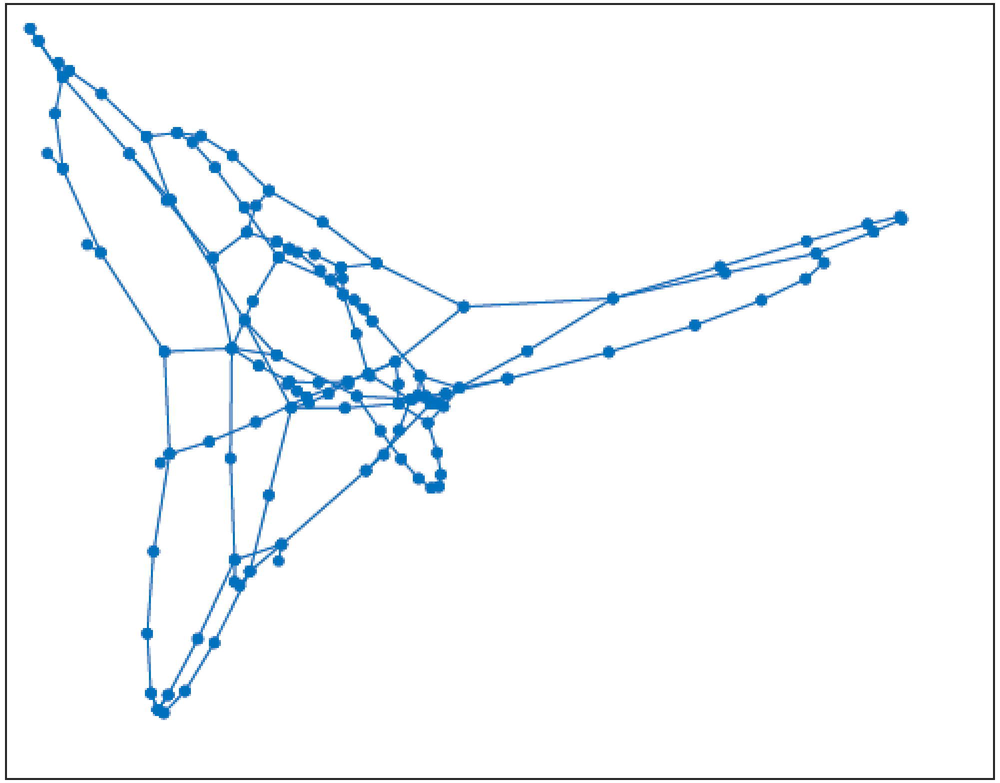


Figure 17. ION Network Topology

Simulation and Results: Scalability (2)

- In terms of throughput, the proposed DGN approach can meet the required thresholds
- The median value for a gold flow is at around 3300 Kbps, for silver flows at 1300, and for bronze flows at about 600 Kbps, all above the requirements

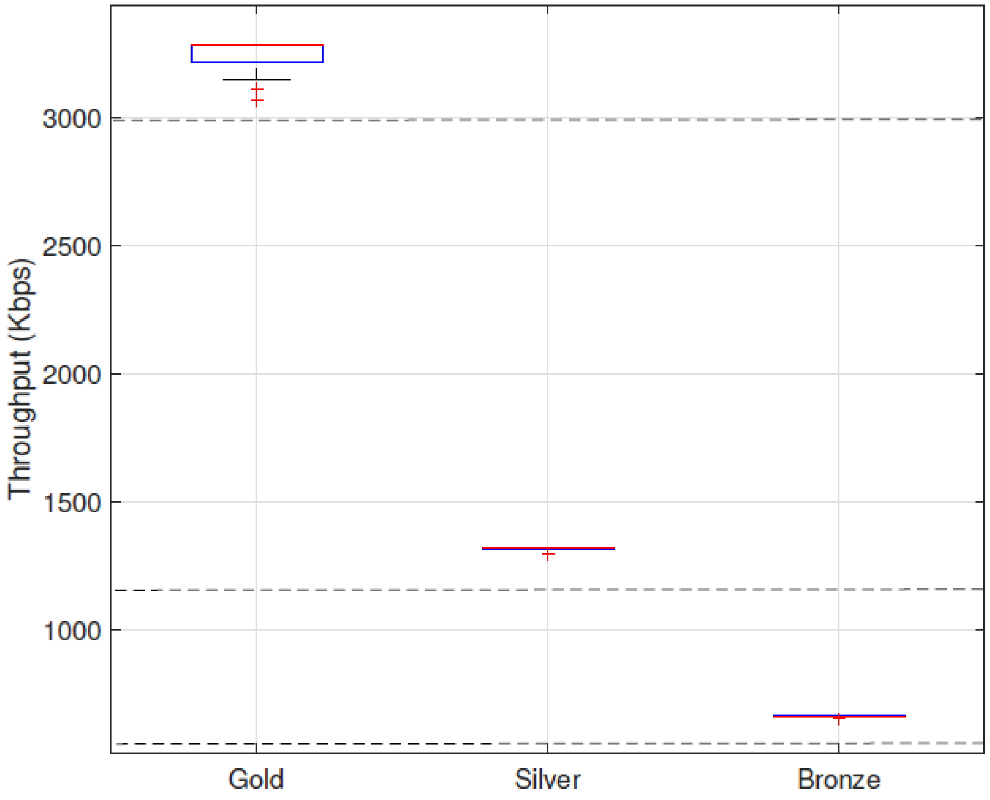


Figure 18. DGN Throughput Result

Simulation and Results: Scalability (2)

- Our DGN proposal can also sustain the flow requirements in terms of end-to-end delay
- The maximum delay values sit at around 0.18, 0.45, and 0.88 seconds for the gold, silver and bronze flows, respectively
- These are below the required thresholds of 0.2, 0.5, and 0.9 seconds for gold, silver, and bronze, respectively

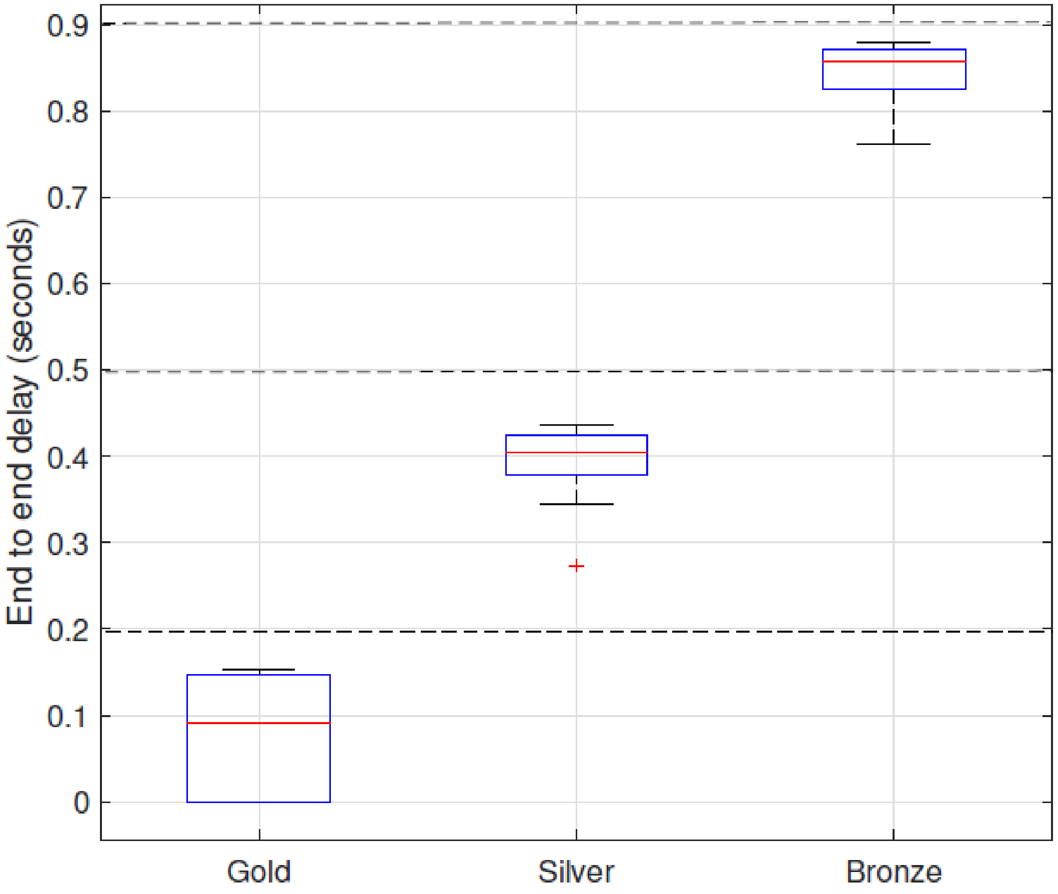


Figure 19. DGN Delay Results

Conclusion

- Graph Convolutional Reinforcement Learning for Smart Queue Management
  - We proposed a DGN based multi-agent approach to our smart queuing problem
  - The agents are tasked with continuously setting the weights for a WFQ algorithm place at ingress nodes
  - The objective is to meet SLA requirements for a set of classified network flow groups
  - These groups are in descending order of priorities: gold, silver and bronze
  - Agent communication is limited to neighbourhood and DGN can learn how best to communicate
- Multi-Agent Deep Q-Learning
  - As benchmarks, we propose two DQN based approaches to the same problem
  - The first is completely distributed with no inter-agent communication or cooperation
  - The second is fully centralized with the agents having the same states, observation and rewards while acting as a unit on the environment

Conclusion

- Results
  - DGN is always able to meet the throughput and end-to-end delay requirements
  - Distributed DQN always violates certain requirements whether in delay or throughput
  - This is due to the lack of any sort of agent communication or cooperation
  - Centralized DQN is however able to meet the required thresholds
  - The centralized, and fully cooperative in a sense, nature of this DGN implementation makes up for the former
  - Our learning approaches prove to be more efficient in handling the problem than classic approaches like PQ

Conclusion

❑ Future avenues to explore

- Multi-layer architectures and the significance of the convolutional layers on agent communications
- Different neighbourhood requirements and variable neighbourhood for DGN
- Joint smart queuing – smart load balancing approaches

References

[1] M. Kim and B. Eng, “Deep reinforcement learning based active queue management for iot networks,” PhD Thesis, 2019.

[2] M. M. Roselló, “Multi-path scheduling with deep reinforcement learning,” in 2019 European Conference on Networks and Communications (EuCNC), pp. 400–405, IEEE, 2019.

[3] M. Guo, Q. Guan, W. Chen, F. Ji, and Z. Peng, “Delay-optimal scheduling for heavy-tailed and light-tailed flows via reinforcement learning,” in 2018 IEEE International Conference on Communication Systems (ICCS), pp. 292–296, IEEE, 2018.

[4] V. Balasubramanian, M. Aloqaily, O. Tunde-Onadele, Z. Yang, and M. Reisslein, “Reinforcing cloud environments via index policy for bursty workloads,” in NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–7, IEEE, 2020.

[5] B. Liao, G. Zhang, Z. Diao, and G. Xie, “Precise and adaptable: Leveraging deep reinforcement learning for gap-based multipath scheduler,” in 2020 IFIP Networking Conference (Networking), pp. 154–162, IEEE, 2020.

[6] M. Bachl, J. Fabini, and T. Zseby, “Lfq: Online learning of per-flow queuing policies using deep reinforcement learning,” in 2020 IEEE 45th Conference on Local Computer Networks (LCN), pp. 417–420, 2020.