International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

Software-Defined Networking-Based Campus Networks Via Deep Reinforcement Learning Algorithms: The Case of University of Technology

Ali Sadiq Salim and Ameer Mosa Al-Sadi
Department of Computer Engineering, University of Technology- Iraq, Baghdad, Iraq.

Abstract: As a consequence of the COVID-19 pandemic, networks need to be adopted to satisfy the new situation. People have been introduced to new modes of working from home, attending teleconferences, and taking part in e-learning. Other technologies, including smart cities, the Internet of Things, and simulation tools, have also seen a rise in demand. In the new situation, the network most affected is the campus network. Fortunately, a powerful and flexible network model called the software-defined network (SDN) is currently being standardized. SDN can significantly improve the performance of campus networks. Consequently, many scholars and experts have focused on enhancing campus networks via SDN technology.

Integrating deep reinforcement learning (DRL) with SDN is pivotal for advancing the quality of service (QoS) of contemporary networks. Their integration enables real-time collaboration, intelligent decision making, and optimized traffic flow and resource allocation.

The system proposed in this research is a DRL algorithm applied to a campus network—the University of Technology—and investigated as a case study. The proposed system employs a two-method approach for optimizing the QoS of a network. First, the system classifies service types and directs TCP traffic by using a deep Q-network (DQN) for intelligent routing; then, UDP traffic is managed using the Dijkstra algorithm for shortest-path selection. This hybrid model leverages the strengths of machine learning and classical algorithms to ensure efficient resource allocation and high-quality data transmission. The system combines the adaptability of DQN with the proven reliability of the Dijkstra algorithm to enhance dynamically the network performance.

The proposed hybrid system, which used DQN for TCP traffic and the Dijkstra algorithm for UDP traffic, was benchmarked against two other algorithms. The first algorithm was an advanced version of the Dijkstra algorithm that was designed specifically for this study. The second algorithm involved a Q-learning (QL)-based approach. The evaluation metrics included throughput and latency. Tests were conducted under various topologies and load conditions.

The research findings revealed a clear advantage of the hybrid system in complex network topologies under heavy-load conditions. The throughput of the proposed system was 30% higher than the advanced Dijkstra and QL algorithms. The latency benefits were pronounced, with a 50% improvement over the competing algorithms.

Introduction

The impact of the COVID-19 pandemic on education has been widespread, with many schools and universities being forced to close because of health issues [1]. This scenario has led to learning disruptions that urgently need alternative solutions. One solution is the use of communication technology, such as the Internet and networks, to facilitate online or digital learning systems. However, the massive increase in demand for these systems has strained their quality of service (QoS), such as those involving network bandwidth. Insufficient bandwidths negatively affect the quality of video transmission and data transfer [2]. In consideration of these challenges, researchers have focused on enhancing the performance of campus networks and improving QoS.

The campus network is a crucial part of the infrastructure of a university because it provides high-speed, reliable, and secure network access to students, faculty, and staff. The network must handle many users accessing it at any single time while managing the growing demands of new technologies, such as cloud computing, big data analytics, and Internet of Things (IoT) devices. Meeting these demands requires campus networks to be highly scalable, flexible, and adaptable, with the ability to add or remove network components as required. Software-defined networks (SDNs) have emerged as a promising solution to enhance the performance and security of campus networks, especially since they can be programmed and managed without manually configuring hardware [3].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

SDNs represent a networking paradigm that allows the centralized control and management of network devices and traffic. In an SDN architecture, the control plane deciding the network behavior and configuration is detached from the data plane, which physically forwards network traffic (Figure 1). This separation enables administrators to program and control the network by using software instead of manual configuration. SDNs provide high flexibility, scalability, and automation in network management. By reducing the need for manual configuration and the time and resources required to debug and integrate new devices, SDNs help ease the limitations of traditional campus networks while improving their overall performance [4]. Recently, SDN technology has become widely popular in network development research. SDN programmability implies efficient network application, bringing application benefits to the entire network—from providers to end users [5, 6]. SDNs are used to produce a logical view of the entire network (i.e., a network abstract, which is the responsibility of the control plane; meanwhile, traffic forwarding is managed at the data plane [7].

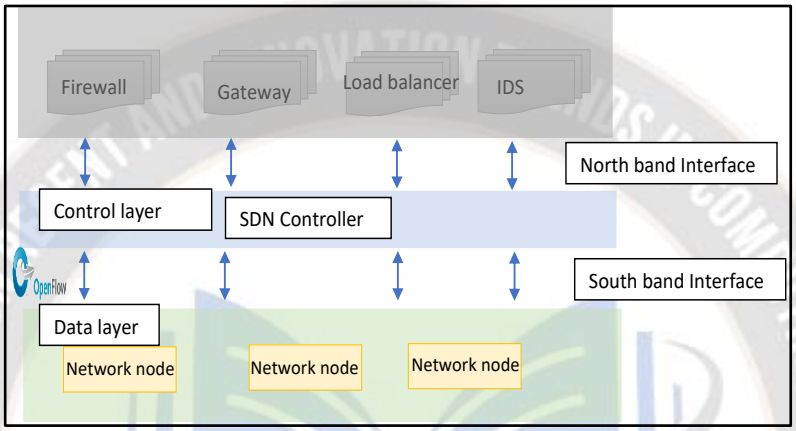


Figure 1. SDN architecture

The SDN technology entails several challenges and limitations when applied to campus networks. For instance, security issues in SDNs involve the centralization of network control and management of network traffic. Centralization opens networks to attacks, as a single point of failure may considerably affect the entire network. Therefore, a secure SDN infrastructure and the enforcement of proper policies regarding network control must be ensured [8]. Experts and researchers classify the security challenges in SDNs into two main categories: infrastructure security of the SDN and enforcement of security policies pertaining to network control. Thus, security also involves ensuring the confidentiality, integrity, and availability of the network and protecting users against attacks, such as network intrusion, data tampering, and unauthorized access [9]. A variety of solutions have been proposed to address security challenges, such as implementing encryption, access control, and firewalls. Virtualization and protocols also help enhance the security of SDNs. The other challenges and limitations are as follows:

- Controller design [10]: Reliability, scalability, and integrity cannot be enhanced by simply using a single controller. Multiple controllers need to be distributed physically.
- SDN implementation in traditional networks [11–13]: One complicated issue is knowing what old network devices could not support the SDN protocol and further knowing how to deal with the interdomain routing protocol problem.

However, SDNs offer a variety of benefits.

- Cost effectiveness [14]: Apart from network devices that are easily optimized, SDNs reduce operational costs.
- Data traffic control [15]: This protocol is the most important advantage of SDNs. In contrast to traditional networks, data traffic control in SDN can more easily direct data traffic, and QoS for video streaming and audio transmission is much easier to employ.
- Management [16, 17]: The management of network resources in SDNs, which takes a global view of the entire network, is much easier than that of traditional networks.

A comprehensive comparison between SDN and the traditional network is shown in Table 1 and Figure 2 [18, 19].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

Table 1. Comparison between an SDN and the traditional network

Criteria	Traditional	SDN
Network management	Difficult because it is managed manually	Easier because it is managed automatically with the help of a controller
Network control	Performed for each device and each packet, such as the routing algorithm	Performed by a controller at each flow
Production of a global view of the network	Difficult because each device employs decision individually	Easier because the controller has a global view of the network
Maintenance cost	Higher because it needs more human resources and time	Less because the controller applies SW runs
Authenticity, integrity, and consistency of the forwarding table and network state	More complicated than those of other devices (routers) along the packet path	Less complicated because it is applied by a controller only
Resource utilization	Less because of its restricted policy	High because of its full dynamic behavior

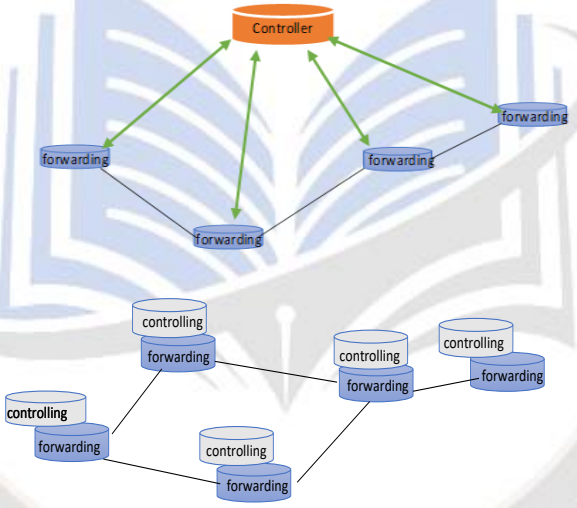


Figure 2. Comparison between the traditional network and SDN

2.5 QL versus deep Q-learning (DQL)

The Q-value function Q(s,a) in QL represents the expected cumulative reward of taking action *a* in state *s*. Then, the optimal policy is implemented.

The key equation in QL is the Bellman equation for Q-values, which is given by

$Q(s,a)=r+\gamma \max_{a'} Q(s',a')$,

where *s* and *a* are the current state and action; ′s′ is the subsequent state after taking action *a* in state *s*; *r* is the immediate reward; and γ is the discount factor.

DQN model

In DQN, the Q-value function is approximated using a neural network. Let us denote this neural network function as $Q(s,a;\theta)$, where θ represents the parameters (or weights) of the network.

Loss function: The aim of DQN is to minimize the mean-squared error between the predicted Q-value and the target Q-value. The loss function L is defined as

$$L(\theta)=E(s,a,r,s')\sim U(D)[(r+\gamma \max a' Q(s',a';\theta-)-Q(s,a;\theta))2],$$

where D is the replay buffer containing the experiences; U(D) shows the uniform sampling from D; and θ represents the parameters of the target network, which is a copy of the main network but is updated less frequently to increase stability.

2.5.1 Reason for using DQN

- Traditional QL in tabular approach: In classical QL, an agent learns from the environment by updating a Q-table containing values for each combination of states and actions. Apart from interacting with the environment, the agent also learns by receiving rewards and updating both the Q-values based on the rewards received and the expected future rewards.
- Limitations of traditional QL: In environments with numerous states and actions, the Q-table is enormous and becomes computationally impractical. This limitation is apparent in complex domains, especially in networks where the state space is extremely large.
- DQL: Large-state spaces are addressed by DQL, which is a combination of QL and deep learning. In this approach, a neural network approximates the Q-value function and takes the current state as input and outputting Q-values for all actions.
- Replay memory: DQL introduces replay memory to address issues concerning sequence–experience correlations. The interactions (state, action, reward, and next state) of an agent are stored in the replay memory. During training, random batches of these experiences are sampled to update the network. This scheme not only disrupts the risky correlations but also allows the algorithm to learn from individual experiences many times.
- Advantages of using networks: In networking scenarios requiring frequent and quick decision making, the traditional tabular QL approach is inefficient. DQL allows for a much faster decision making because the Q-values for all actions in a given state are calculated with a single forward pass through the neural network. This approach considerably speeds up the decision-making process and further improves network performance, especially in dynamic environments.
- Conclusion: By adopting DQL in networking scenarios, users can harness deep learning to ensure fast and informed decisions without requiring exhaustive Q-table computations. This approach is beneficial in situations in which the network needs to adapt to changing conditions, such as when using SDNs or network routing protocols that allow varying traffic conditions to be adjusted.

2.6 Related work for SDN and Reinforcement learning

Chavula et al. [20] adopted a technique to enhance bandwidth utilization and reduce latency in SDN by using QL for path selection. Their method considered link latency, link bandwidth, and available link bandwidth and could be tested on a partial mesh network. Their results showed increased multipath throughput and prioritized latency to reduce latencies. As both latency and bandwidth were considered, the performance was similar to that of standard approaches.

Kim et al. [21] examined a solution to address congestion caused by the Dijkstra shortest path, which ignores account bandwidth. In SDNs, QL is used to select a new path once the congestion threshold on the original path is reached. The method was tested on a simulated partial mesh network. The transmission time of the examined solution was slightly faster than those of the Dijkstra shortest-path and the extended shortest-path methods.

Adaptive Q-routing full echo (AQRFE) [22] is a variation of QR that integrates a dynamic learning rate to each node apart from the standard learning rate. The goal is to improve QR exploration by using the standard learning rate when the Q-value originates from a neighboring node and a new learning rate in other cases. Previously, AQRFE was tested on a partial mesh network and compared with QR and dual QR. AQRFE had a low average delivery time in the low-load network and performed similarly to QR and dual QR in the high-load network.

Dual reinforcement Q-routing (DRQR) [23] calculates paths handling high-load levels with a low-average delivery time. In QR, when node A requests latency estimates to the destination from all its neighboring nodes, each neighbor sends back this information. Then, node A uses the estimated shortest path in the reward formula to update its Q-table, a process known as “forward exploration.” DRQR adds “backward exploration.” This scheme enables node A to share information about the traversed path with its neighboring nodes so they can update their own Q-tables. Previously, DRQR was tested on a partial mesh network and compared with the shortest-path approach and QR. The shortest-path approach obtained the lowest average delivery time at low loads. DRQR performed better than QR at low loads and better than both the shortest-path approach and QR at moderate and high loads.

Sendra et al. [24] proposed a routing optimization strategy for SDN, in which RL was used to improve the QoS of a network. Their approach involved an RL agent selecting the optimal paths with the lowest cost. Three parameters were considered: delay, packet loss rate, and bandwidth.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

Ref. [25] proposed the use of RL to meet QoS requirements. An RL-driven QoS-aware routing algorithm included a QoS monitoring module to compute the delay and packet loss ratio and an RL-based intelligent routing decision-making (RIRD) module to interact with the monitoring system. Then, the best path was selected. The RL agent obtained the highest reward value when the route with the minimum delay and packet loss ratio was selected.

This study investigated the architecture and components of a campus network by taking the University of Technology (UOT) as the case study. Ref. [26] introduced QR-SDN, a model based on classical tabular reinforcement learning and aimed at optimizing flow routing in SDNs. In contrast to other approaches, QR-SDN enables flow-preserving multipath routing by directly representing flow routes in its state-action space. The evaluation of QR-SDN revealed substantially lower latencies compared with the traditional single-path approach. However, the system was challenged by scalability issues when the number of network nodes increased. This limitation opens various future research directions, including scalability improvement, development of novel exploration strategies, and incorporating deep reinforcement learning (DRL) to address routing in SDNs.

Designing the proposed method

1.1 Architecture of the proposed system

This study explored the architecture of the proposed system and determined the routing traffic by using reinforcement learning (Figure 3).

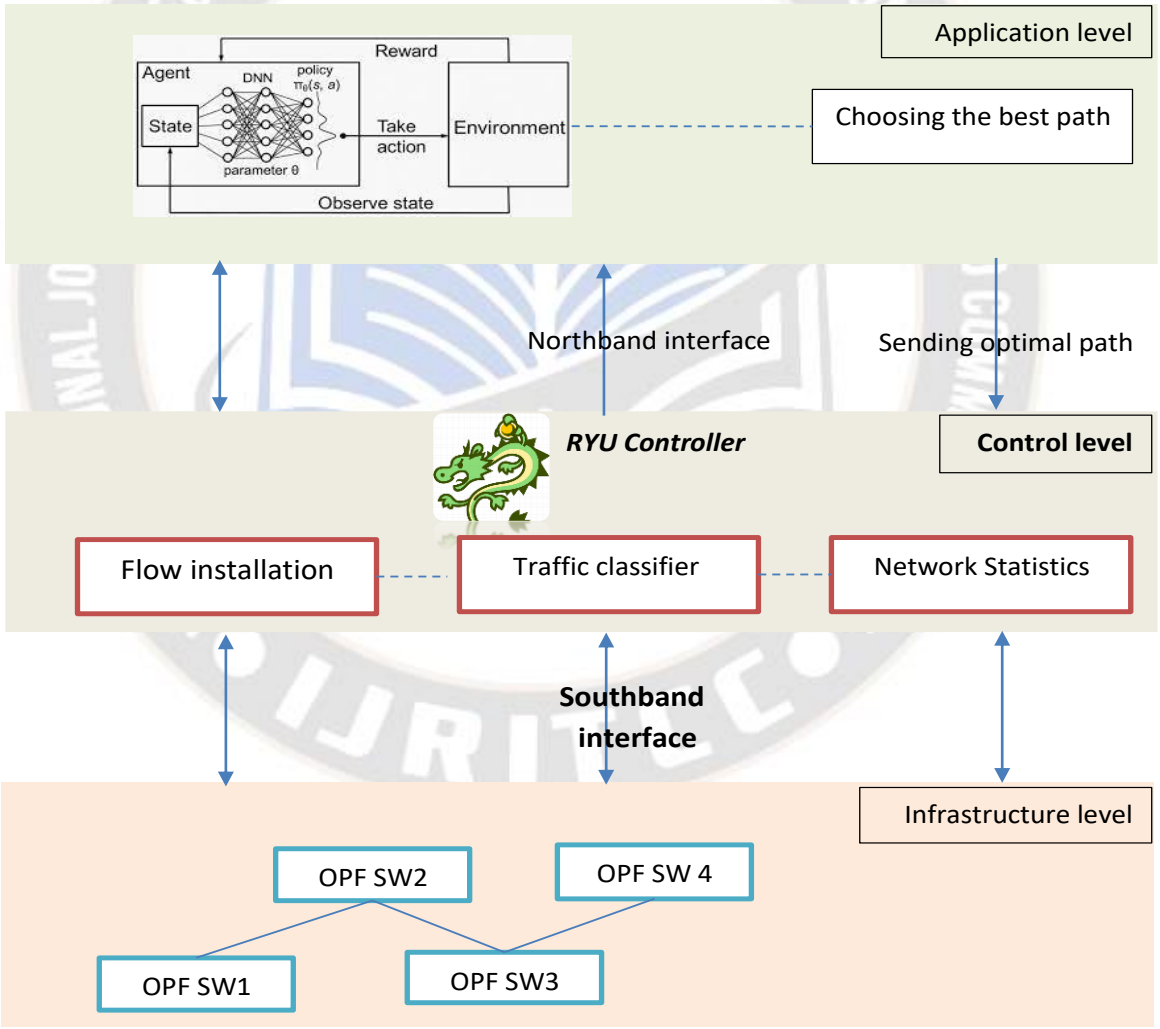


Figure 3. DRL-based architecture of traffic routing for campus networks

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

3.2.1. Infrastructure plane

The infrastructure plane comprises three primary components: the providers of the service (server), the SDN-based networks, and the clients or users benefitting from the services of the server. This plane is connected to the control plan via the southbound interface. The network comprises the forwarding switches managed by the controller and the media (links) for switch connection. On the basis of the flow table created by the controller, the switches forward or drop packets. The SDN switches do not have any knowledge of the network and merely rely on the control and application planes that configure their respective flow tables.

The control plane periodically requests updates from the infrastructure plane regarding the topology of the network. Therefore, the infrastructure plane is also regarded as a reinforcement learning environment.

3.2.2. Control plane

The control plane has an abstract view (i.e., a global view) of the whole network. The management of the network is the responsibility of this plane. This plane includes different modules with a variety of responsibilities. The first module is "network knowledge," which represents the topology as a graph and stores physical information. The second module is flow tracking while introducing statistical information for the network. The third module is the service classifier, which prioritizes TCP flows over UDP. In the fourth module, the flow is installed, and flow entries are built into the switches. The fifth module is for the data computation of the QoS parameters. The sixth module is for storing the QoS metrics and keeping records of the source and destination nodes related to the QoS metrics.

3.2.3. Application layer

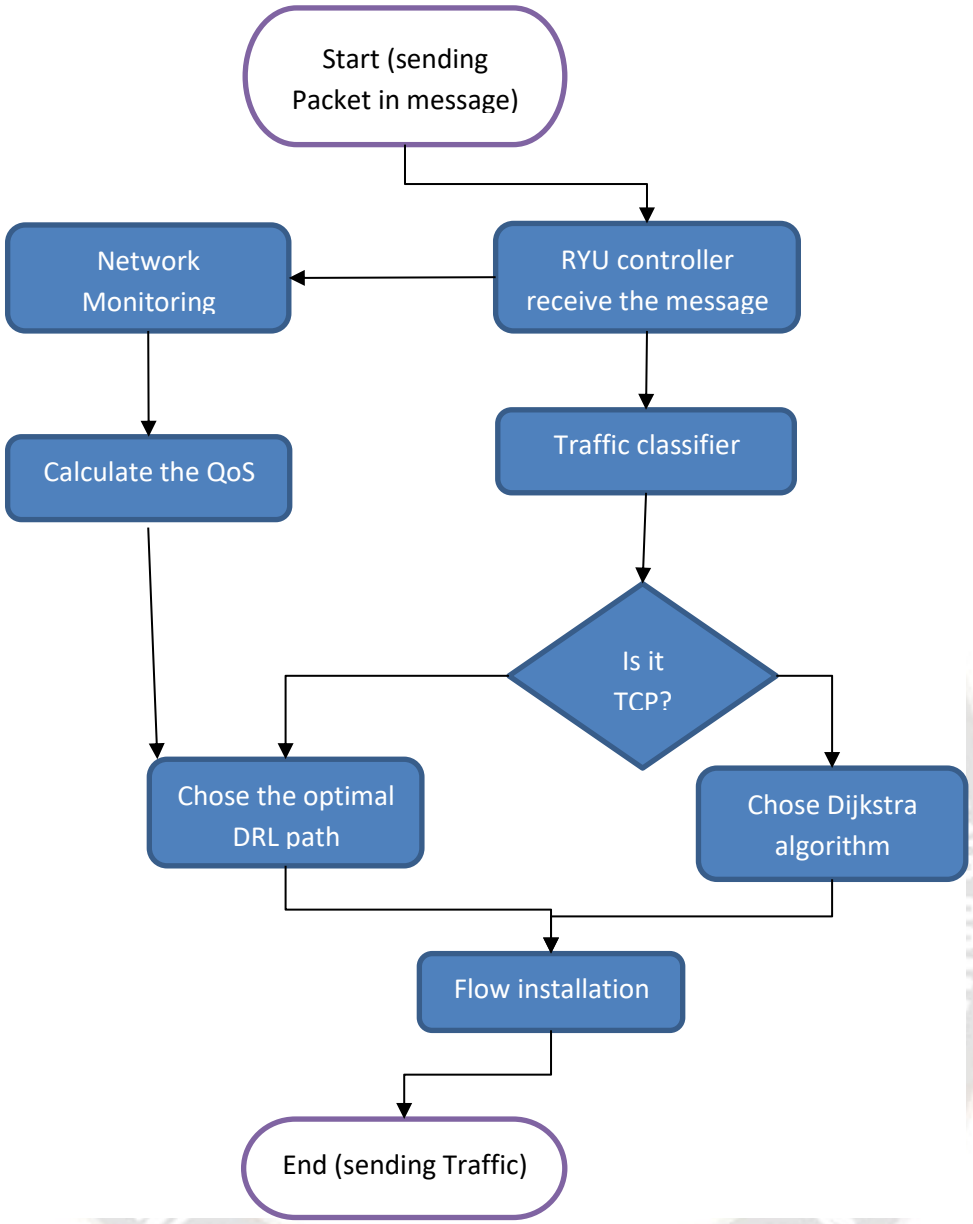
The application layer manages the reinforcement learning of network characteristics and makes intelligent decisions about route calculation. All information from the control plane required to compute the optimal route is addressed in the northbound interface.

The RL method communicates with the control plane via the northbound interface.

3.3. Processing of the proposed system

When the client host requests a service (e.g., connection, the first module) from the server host, a packet-in message is initially sent to the controller. The controller monitors the topology of the network and checks the packet type by using a "service classifier." This classifier checks the packet and evaluates its best route. Then, the last two modules (data computation and QoS storage) are in the control plane extract and record the QoS metrics. Finally, all information is sent to the application plane for optimal routing, and the flow builder installs the flow entry into the flow table. The flowchart of the process is shown below.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*



DRL-based decision-making solutions

4.1. Problem domain

Graph theory is used to represent the infrastructure plane as an undirected graph G(N, K), where N is the set of nodes, and K is the set of links between the nodes. The set of nodes N is divided into four subsets: C, V, W, and T. C represents the clients, V represents the servers, W represents the OpenFlow switches, and C represents the SDN controller. The graph model shows the interactions of the elements in the infrastructure plane. Each client c, which is a member of the set C, is connected to a forwarding switch w, which is a member of set W. The connection is implemented through link k, which is a member of set K. Each link in the network has a limited capacity Ck. This capacity limits the number of flows that can be passed through the network. Each traffic flow f is a member of a set of flows F. The strategy primarily centers on the effective selection of an appropriate path when clients make requests for a service at any given time. R represents the set of potential routes, and the routing method determines the most workable path r, which is a member of R. in this study, a path is defined as a set of links r = k1, k2,..., kn, which represents the route between the client and server. The optimization method proposed in this study incorporates a network monitor for periodically monitoring the network status and making path decisions based on the current status of the network links. The DQN agent rapidly reacts to congestion on a link and suggests alternative paths for the service. The proposed system considers two link-state metrics for each path r: available bandwidth Bk and delay Dk. The metrics are used to evaluate the quality of each path and determine the most

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 25 August 2023 **Revised:** 14 October 2023 **Accepted:** 28 October 2023*

optimal. By considering these two link-state metrics, the proposed solution aims to provide an optimal viewing experience for the clients.

Strategy to calculate the QoS metrics

Available bandwidth

The amount of bandwidth routed between the source and destination is determined via the control plane, which monitors traffic flow and periodically gathers the received and transmitted bytes at each port. Therefore, the used bandwidth can be measured via OFPortStatsReply messages (i.e., the amount of received bytes at a specific time) at two different times. The used bandwidth is calculated as

$BW_{used} = (BW_{received\ t2} - BW_{received\ t1}) / \Delta t.$ (1)

This calculation represents the amount of bandwidth used over a certain period.

The available bandwidth of link k is calculated as

$BW_{available} = Ck - BW_{used}.$ (2)

This calculation subtracts the used bandwidth from the total link capacity to determine the amount of bandwidth that is available for a particular link.

Link delay

Delay affects QoS, particularly at the beginning of the viewing experience, resulting in a start-up delay. Therefore, delay Dk should also be considered when determining the optimal path for campus networks. The measurement of Dl is elaborated in Ref. [36]. Briefly, the delay in the link between the source switch and the destination switch ($Dw_{src}w_{des}$) must be determined. First, the total delay from the controller to the source switch and then to the destination switch is calculated. This calculating path is implemented by sending an LLDP packet, in which the difference time (sending time and receiving time) is measured. Then, the delay between the controller and switch (source and destination) and the OFPortEchoRequest and OFPortEchoReply, which contain sending time t_{send} and receiving time t_{rcv} between the controller and any switch, respectively, are calculated. The delay in the link between the source switch and destination switch is calculated as

DZy = ST - RT 2. (3)

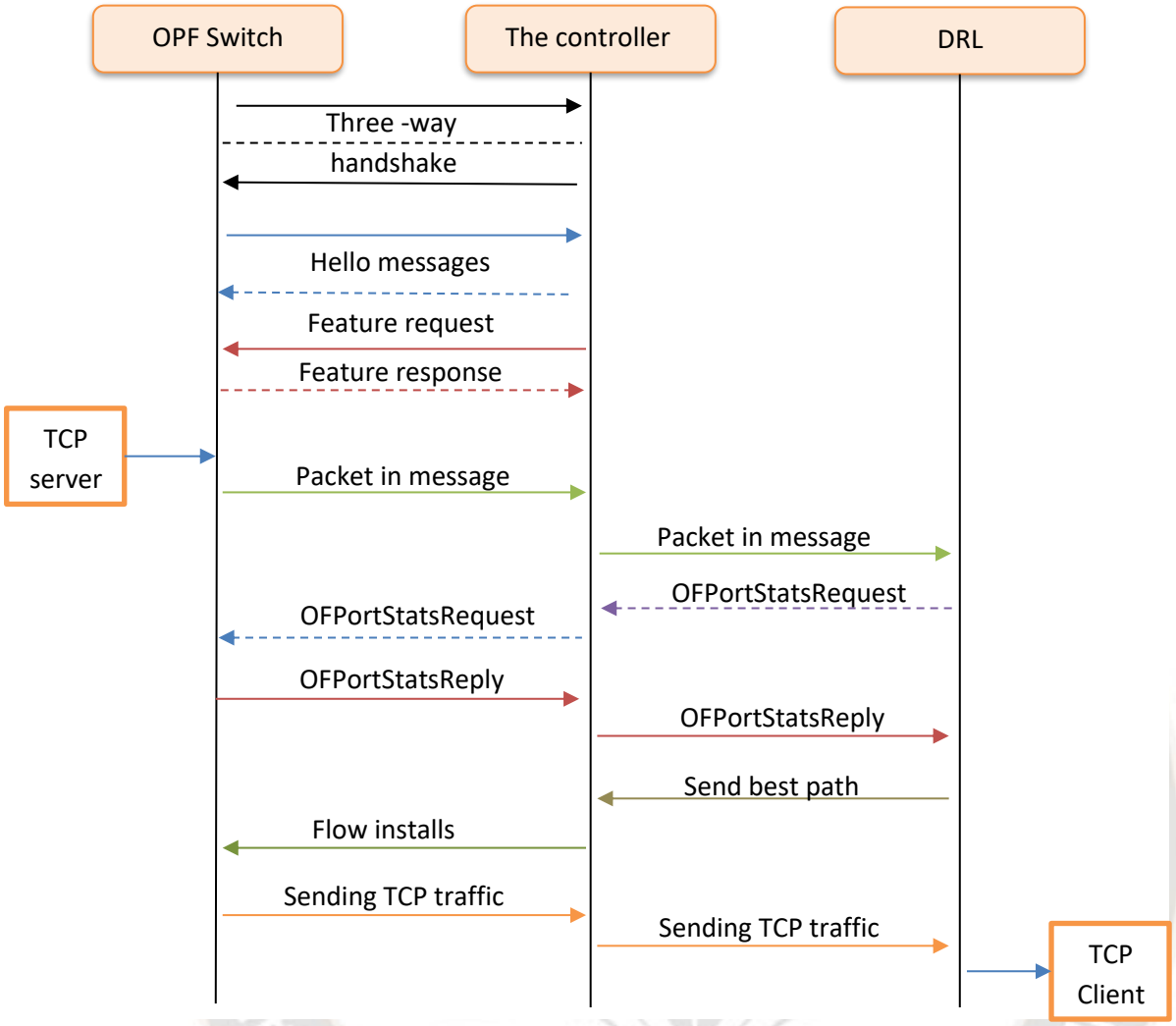
Now, the calculation of the delay between y1 and y2 is presented as follows:

Dly1,y2 = Td - DZy1 - DZy2 (4)

Figure 3 shows the message exchange between the server and the client. After negotiation messages are completed between the SDN switch and controller, and after the server transmits packets, the SDN switch sends packet_In to the controller. The general overview of the controller uses statistical information to make routing decisions. It also monitors the network and further extracts statistical information, which is used to compute the optimal path for traffic between the two pairs. The flow installation module installs this path in the network, allowing traffic to be transmitted along the optimal route. The goal of DRL-based campus networking is selecting the optimal path between the aforementioned two pairs passing through the traffic. This selected path avoids high delay and jitter and involves maximum available bandwidth to enhance the QoS of the campus network.

Sequence diagram of OpenFlow with RYU

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*



4.2. RL-based solution

The RL agent is a system that learns decision-making rules by interacting with its environment and observing the results of its actions. This scheme is achieved by exploring a variety of options and evaluating their outcomes. Information is used to determine the best strategy for achieving the goal of the agent, which is to identify the most efficient path for traffic in an SDN. Thus, the throughput for users (i.e., their QoS requirements) is also maximized. The RL agent uses a DQN algorithm to solve the optimization problem. The proposed problem is grouped into many parts, as described in the succeeding sections.

1. State space S

The state space captures all network configurations that may be encountered. It combines different factors, such as available bandwidth, link delay, and current node.

$st=[nt,BW,Dk]$,

where nt is the current node at time t; Bt is the available bandwidth at node nt at time t; and Dk is the link delay at node nt at time t.

2. Action space A

The action space represents all actions (choices) of the next node to move to and from the current node.

$at \in N(nt)$,

where $N(nt)$ is the set of neighboring nodes to the current node nt.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 25 August 2023 **Revised:** 14 October 2023 **Accepted:** 28 October 2023*

3. Reward function *R*(*s*,*a*)

The reward function captures the quality of the path based on the QoS parameters of the network.

$R=w1*(1/BW)+w2*$Dk,

where *w*1 and w2 are weights for adjusting the importance of the bandwidth and link delay.

ALGORITHM 1 DQN BASED ROUTING FOR SDN BASED CAMPUS NETWORKS

***Input**: New topology*
Network QoS parameter: Available bandwidth, Packet loss
Learning rate: alpha
Exploration-exploitation parameter: epsilon
Learning episodes number: n_episodes
***Output**: find the optimal path for TCP traffic between source and destination*

```
Initialize DQN and Target DQN networks;
Initialize replay memory D
Initialize Optimizer
for each TCP over (src, dst) in Y do:
    for episode = 1 to n_episodes do
        Initialize state S = src;
        while S != dst do)
            A = choose_action(DQN, S, epsilon);
            R, S' = take_action(S, A, QoS_parameters);
            store_transition(D, S, A, R, S');
            mini_batch = sample(D); mini_batch = sample(D);
            update(DQN, mini_batch, alpha, gamma);
            if episode % some_frequency == 0: , update_target(DQN, Target_DQN);
            S = S';
        end
    end
            epsilon *= epsilon_decay;
end
```

Table 2: Hyperparameter training

Parameter	value
Discount factor	0.99
Learning rate	5e-03
criticOpts.Optimizer	'sgdm'
criticOpts.UseDevice	'cpu';
Starting exploration probability	1.0
Minimum exploration probability	0.05
Epsilon Decay	1e-4
Max Steps Per Episode	1000
trainOpts.StopTrainingCriteria	"AverageReward
trainOpts.StopTrainingValue	550
Dropout to reduce overfitting	0.4
Batch size	64
.MaxEpisodes	= 2000;

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

4 IMPLEMENTATION AND RESULT

4.1 Introduction

This chapter investigates our proposed system's results in different network topologies and applied advanced Dijkstra and QL algorithms in different loads. We will start with the requirement to implement the algorithms, then explain the scenario procedures to make a test, and finally discuss the result for each scenario.

4.2 Requirement

The emulation environments utilized in this study boast the following specifications:

1. This study utilizes the MATLAB programming language, specifically version 2022b, to develop the management algorithm. This algorithm calculates the optimal path between each client-server pairing based on the chosen weight criteria. Representing the controller application, the resultant output of this program is relayed to the controller in the form of a text file. Consequently, the MATLAB application shoulders the responsibility of determining the most efficient path between the clients and servers [62]. In Matalab, create an environment that is compatible with the reinforcement learning tool box, then create a DQN agent and train it in the reinforcement learning tool box.

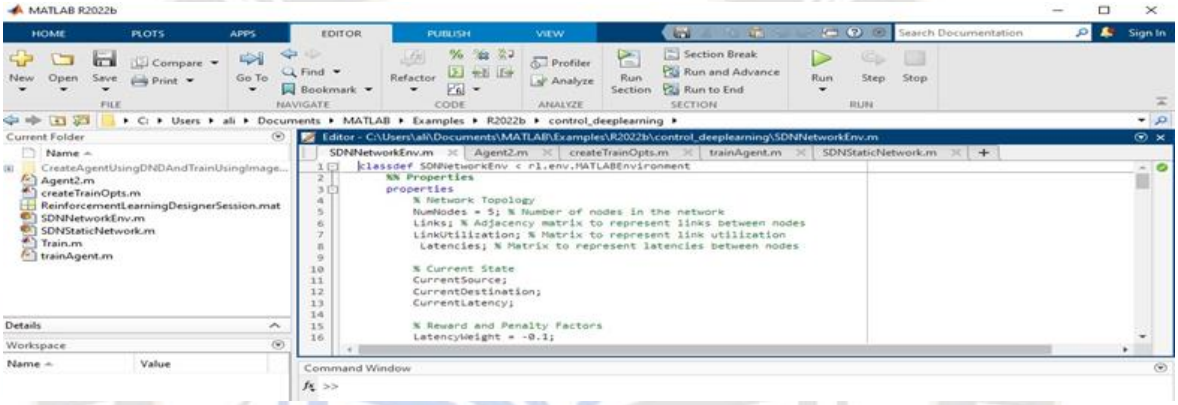


Fig 4.1 Matlab 2022b

2. The Oracle Virtual Machine (VM) VirtualBox version 6.1, a hypervisor, has been installed atop a Windows operating system. Oracle VM VirtualBox is a cross-platform virtualization software that allows a single computer to operate concurrently with multiple operating systems. This includes systems like Microsoft Windows, Mac OS X, Linux, and Oracle Solaris. Given that Mininet operates on the Linux operating system, for the purposes of this research, Linux was installed within VirtualBox version 6.1 to facilitate the setup of Mininet. [63].

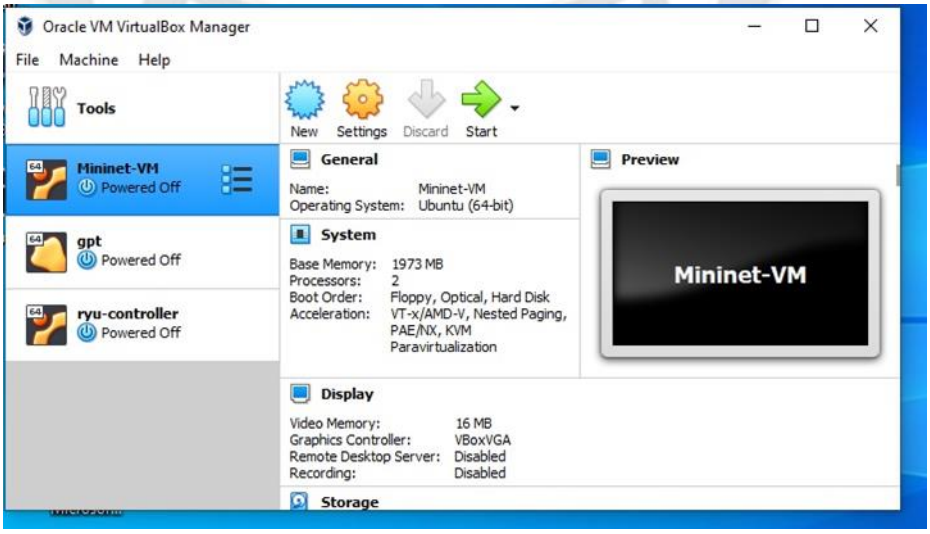


Fig 4.2 Oracle VM VirtualBox

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *14 October 2023* ***Accepted:*** *28 October 2023*

3. The Ubuntu server, version 20.04.1 LTS (64-bit), was configured with the Mininet network emulator version 2.3.0 package. MININET is a network emulator capable of constructing an authentic virtual network on an individual machine by leveraging the actual kernel, switch, and application code, whether on a VM, cloud, or native setup. The switches within MININET support OpenFlow, providing immense adaptability for custom routing and SDN [64]. For the scope of this thesis, version 2.3.0 of MININET has been employed.

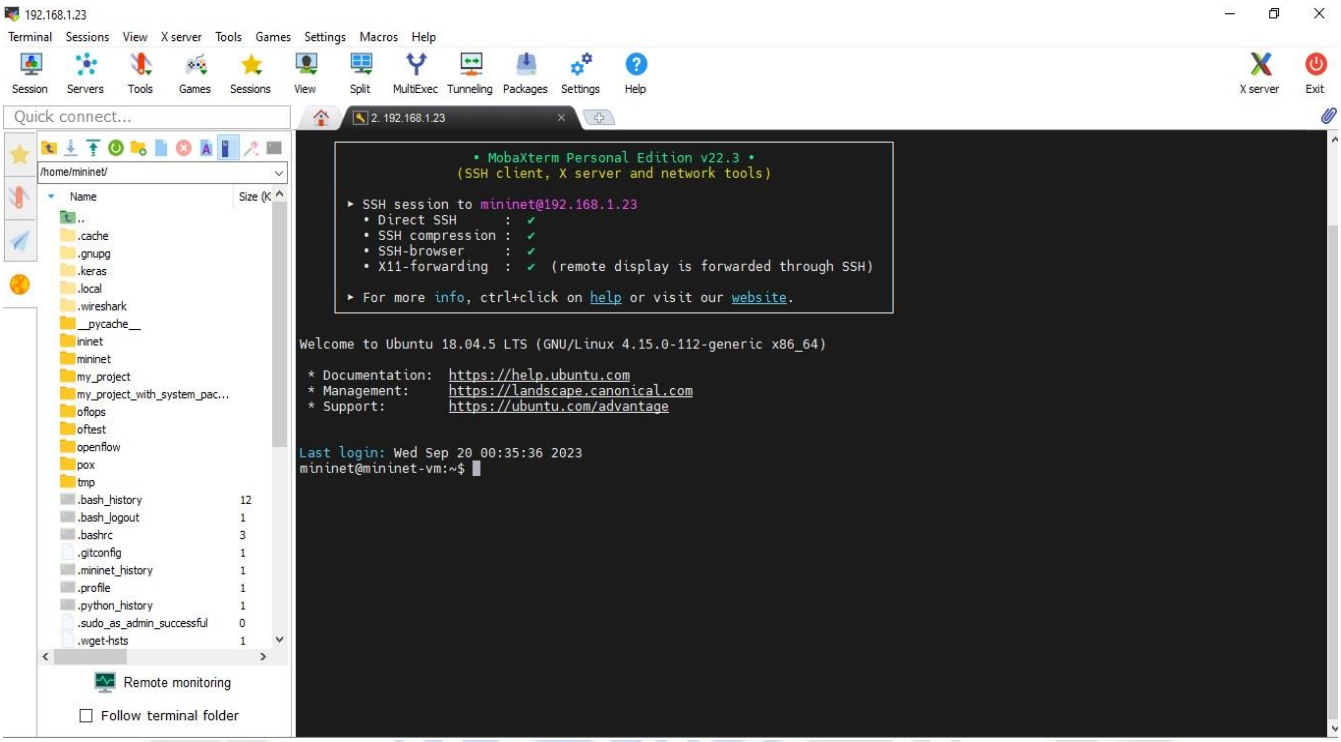


Fig 4.3 Mininet

4. The study employed the SDN RYU controller for network management. RYU is an SDN (Software Defined Networking) controller designed with modularity and flexibility. Originating from Japan, its name means 'flow' in Japanese, aptly reflecting its primary function: managing network flows. Written in Python, RYU provides a cohesive platform for developers to craft networking applications while supporting multiple southbound protocols like OpenFlow. Its intuitive API, active community, and rich set of features make RYU a favorite choice for many looking to delve into the world of SDN

5. Given its comprehensive capabilities, the MobaXterm remote server version 22.0 was selected for remote access to the emulation environment.

MobaXterm is a premier remote server and toolbox designed for remote access and computation. It encompasses an array of vital remote network utilities, including SSH, X11, RDP, VNC, and FTP [65]. In this research, MobaXterm version 22.0 has been employed on the Windows OS to facilitate the creation, modification, and access to Mininet and its associated files.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

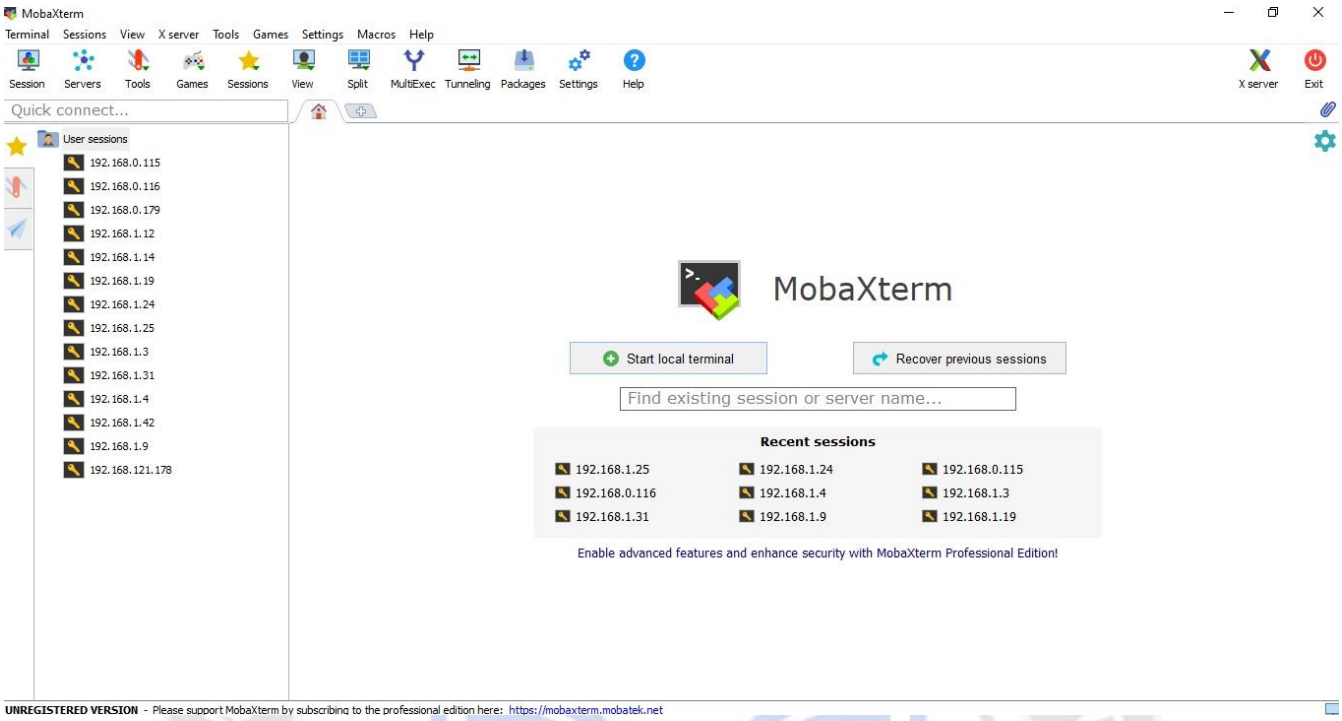


Fig 4.4 MobaXterm

4.3 Implementation Steps

4.3.1 Building and Preparation the Algorithms

1. Build topology in MININET
2. Applied traffic using iperf tool
3. Export the traffic forwarding request (Src-Dst) from RYU controller to the application.
4. Calculate the optimal path and create a decision based on DQN algorithm using (available bandwidth and link delay) for TCP traffic.
5. Or using Dijkstra algorithm for UDP.
6. Import decision to RYU controller to be applied on topology.
7. Measuring the evaluation parameters (BW, latency and jitter).

4.3.2 Implementation of Comparison Between the Proposed System and Other Algorithms

To test our proposed system's performance, we made a series of scenarios. Firstly, we designed and implemented an advanced Dijkstra algorithm with two variable metrics (Available bandwidth and link delay). Also, implementing the Q- learning algorithm in [60]. These algorithms were applied in three network topologies: the first is considered a simple topology containing five nodes, the second topology for medium topology ten nodes and finally, the UOT containing twenty-one nodes. All of the topologies applied in three different loads 0, 300 and 750 Mbps. To test the throughput, jitter and latency, we used different tools to check it.

A) Iperf : that is used to test the bandwidth and jitter.

b) Ping : It measures the latency between the source and the destination. We took an average of 20 timestamps.

To calculate the improvement percentage, we used the below formula

Improvement percentage = ((new value – old value)/old value) *100 …… (6)

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

4.4 Topologies and Result

A) UOT topology

Our recent study examined the network architecture at University of Technology (UOT). We developed a representative network topology comprising 21 nodes using a systematic approach. Each node corresponds to specific departments or operational units within the university, reflecting both its functional and spatial attributes. This topology not only elucidates the intricacies of the university's network framework but also provides a foundational reference for further research in scalable network design. By grounding our study in a real-world institution, we aim to bridge the gap between theoretical network studies and practical applications within the context of higher education infrastructures. The Figures below illustrate the design of UOT in MATLAB, MININET, and the graphical design of the university.

Fig (4.5). The Graphical Design of UOT

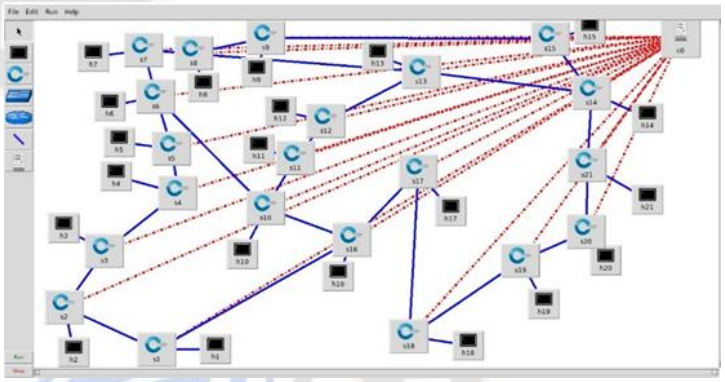


Fig (4.6) Designing The Network Topology of The UOT in MININET

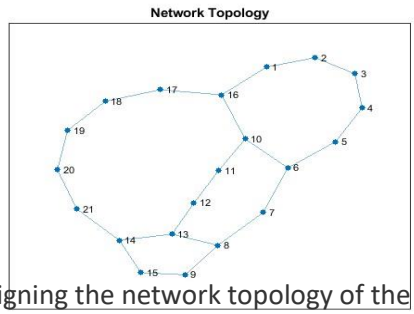


Fig (4.7) Designing the network topology of the UOT in MATLAB

B) Simple topology:

We are creating a simple topology that consists of five nodes with six edges (links), as shown in Figure (4.8) and Figure (4.9). Each link has its own available bandwidth and link delay. The result will be summarized as below:

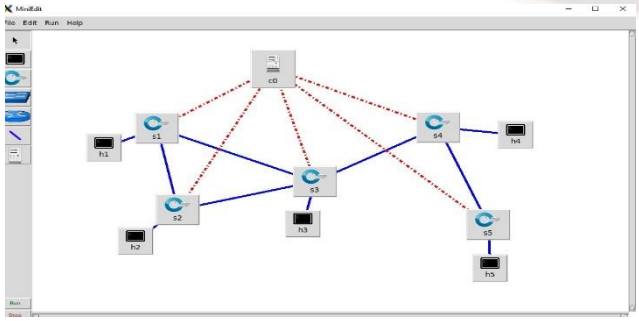


Fig (4.9) Simple Topology in Mininet

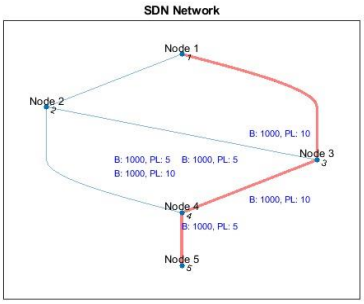


Fig. (4.8) Simple Topo in Matlab

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

C) Medium Topology

We are creating a medium topology consisting of 10 nodes with 15 edges (links), as shown in Figure (4.10) and Figure (4.11) each link has its own available bandwidth and packet loss. The result will be summarized as below:

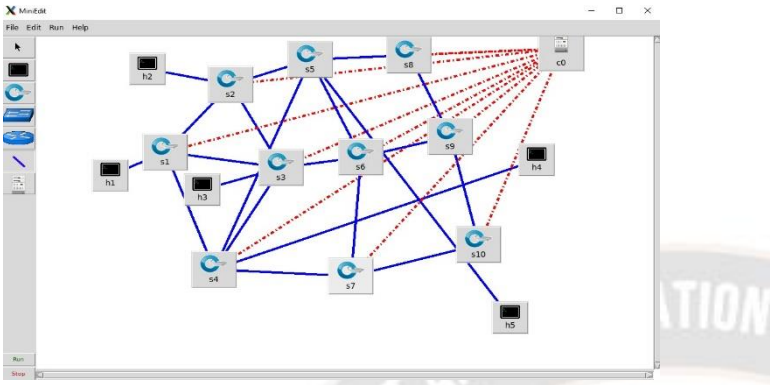


Fig (4.11) Medium Topo with Mininet

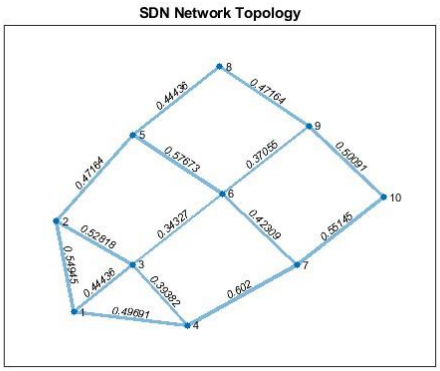


Fig. (4.10) Medium Topo with Matlab

4.5 Experimental Procedure: -

4.5.1) Experiment 1: Advanced Dijkstra algorithm and our proposed algorithm

A) With Simple Topology

1. Figure 4.12 shows the comparison of throughput resulting from our proposed system and the Advanced Dijkstra algorithm; the result shows that there is a little difference in throughput. An Advanced Dijkstra algorithm has better performance in zero and 300 loads, whereas our RL proposed system have a better result in 750 loads.

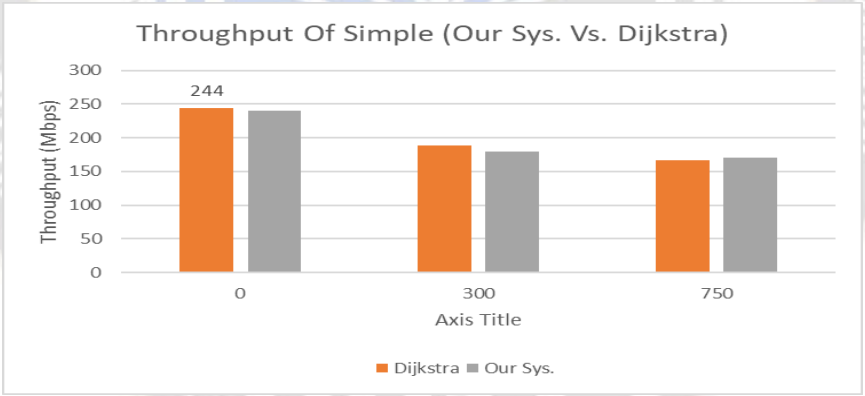


Fig. (4.12) Throughput of Simple Topo (Our Sys Vs. Adv Dijkstra)

2. Figure 4.13 demonstrates, that an Advanced Dijkstra algorithm also has better performance in zero and 300 load with less latency rather than our proposed system, while our proposed system enhances the latency with high load.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

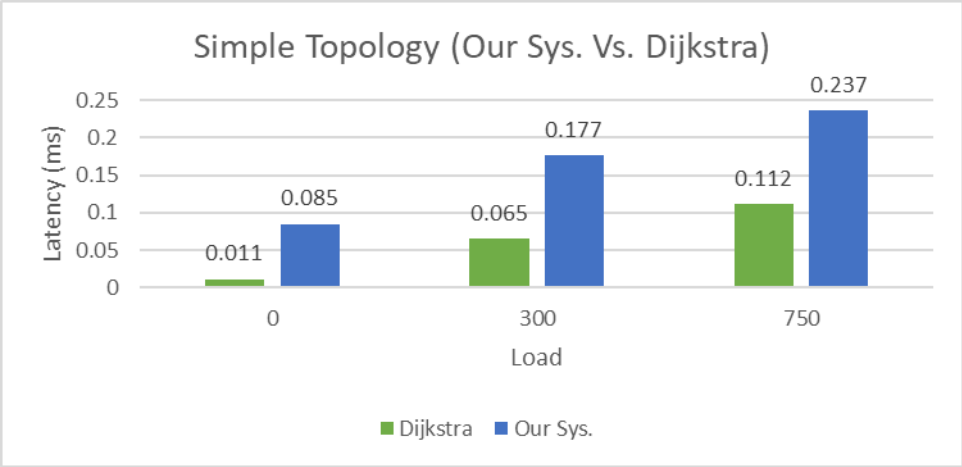


Fig. (4.13) Latency for Simple topology (Our Sys. Vs. Adv Dijkstra)

3. As shown in Figure (4.14), our proposed system has better performance in jitter rather than the Advanced Dijkstra algorithm

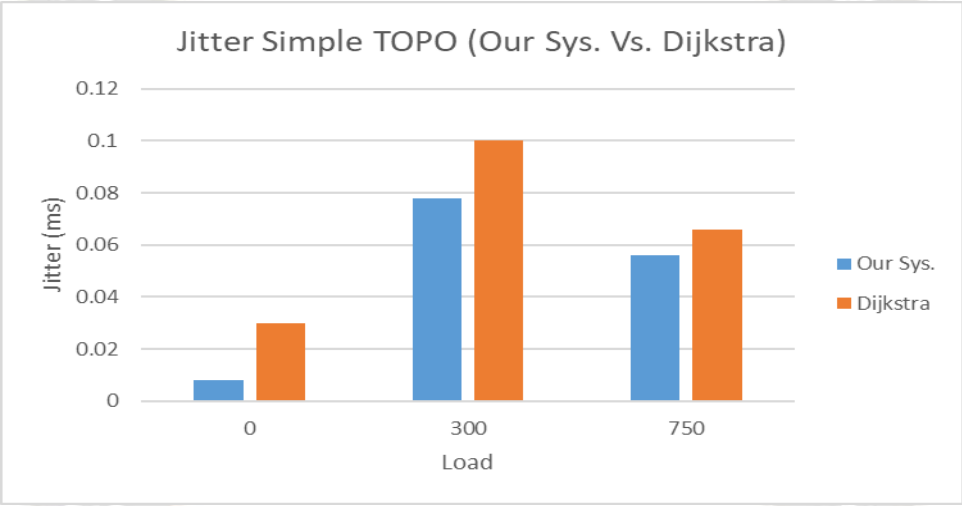


Fig (4.14) Jitter for Simple Topo (Our Sys. Vs. Adv Dijkstra)

B) medium topology

1. As shown in Figure 4.15, our proposed system increases the throughput rather than the Advanced Dijkstra algorithm.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

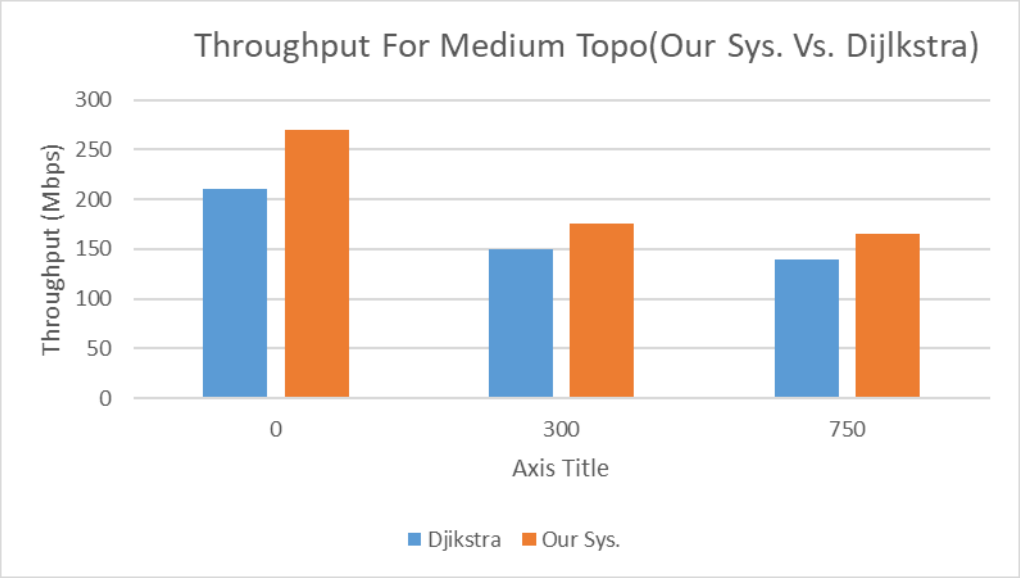


Fig. (4.15) Throughput for Medium topo (Our Sys. Vs. Adv Dijkstra)

The average "difference" in the throughput of the proposed system was approximately 21.25% higher than that of the Dijkstra algorithm at the three load levels

2. Figure 4.16 illustrates our proposed system has also a better performance in all three loads that have less latency rather than the Advanced Dijkstra algorithm, while our proposed system enhanced the latency with high load.

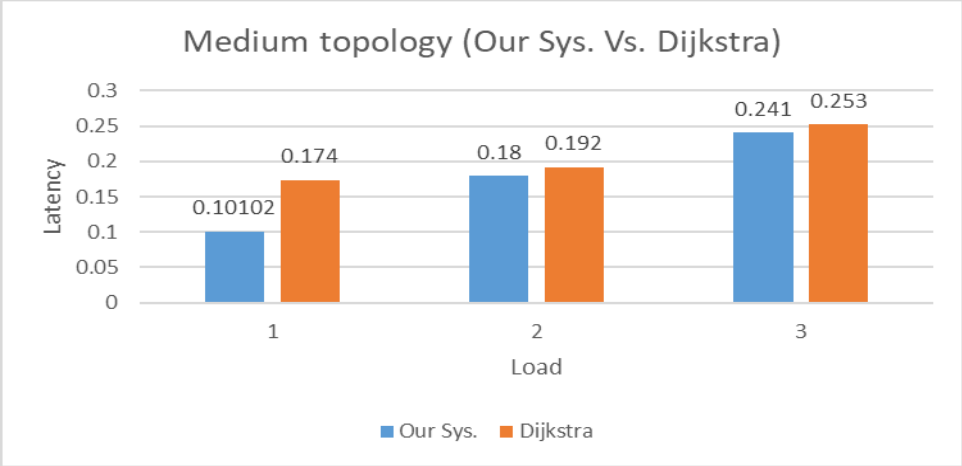


Fig. (4.16) Latency for Medium topo (Our Sys. Vs. Adv Dijkstra)

The proposed system enhanced the latency by approximately 17%, 9%, and 6% at the 0, 300, and 750 load levels, respectively.

3. As shown in Figure 4.17, our proposed system enhances the jitter rather than the Advanced Dijkstra algorithm

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

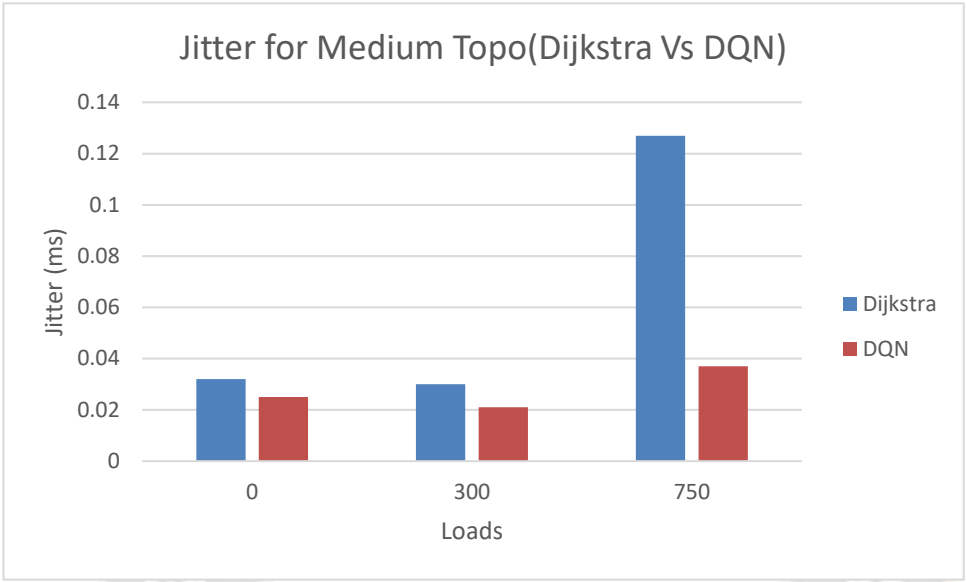


Fig (4.17) Jitter For Medium Topo (Our Sys. Vs. Dijkstra)

C) UOT topology

We are creating a UOT topology that consists of 21 nodes with 24 edges (links), as shown in Figure (4.6). Each link has its own available bandwidth and link delay. The result will be summarized as follows:

1. As shown in Figure 4.18, our proposed system increased the throughput rather than the Advanced Dijkstra algorithm.

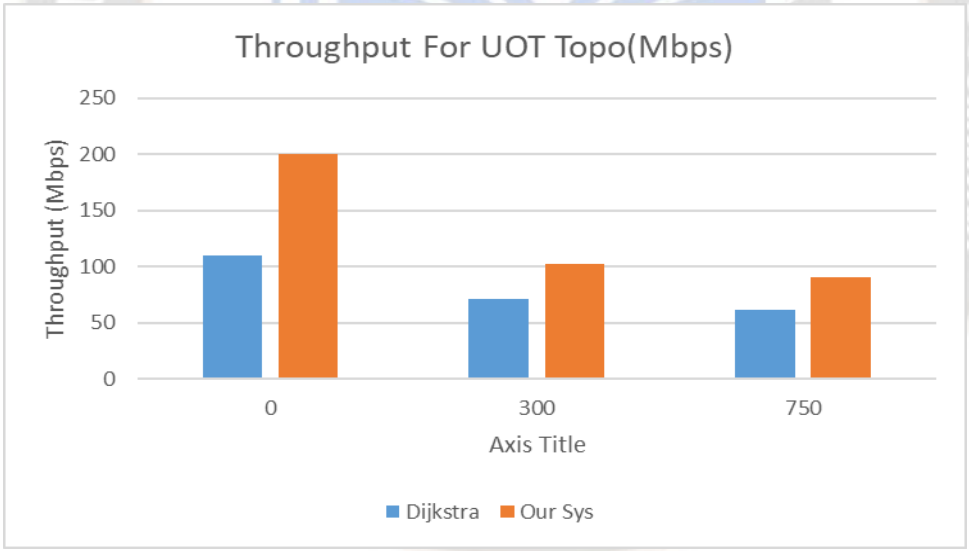


Fig (4.18) Throughput for UOT topo (Our sys. Vs. Adv Dijkstra)

The throughput of the proposed system (82%, 43%, and 93% at the 0, 300, and 750 load levels, respectively) is relatively good for complex networks

2. As shown in Figure 4.19, our proposed system has also a better performance in all three loads that have less latency rather than Advanced Dijkstra

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

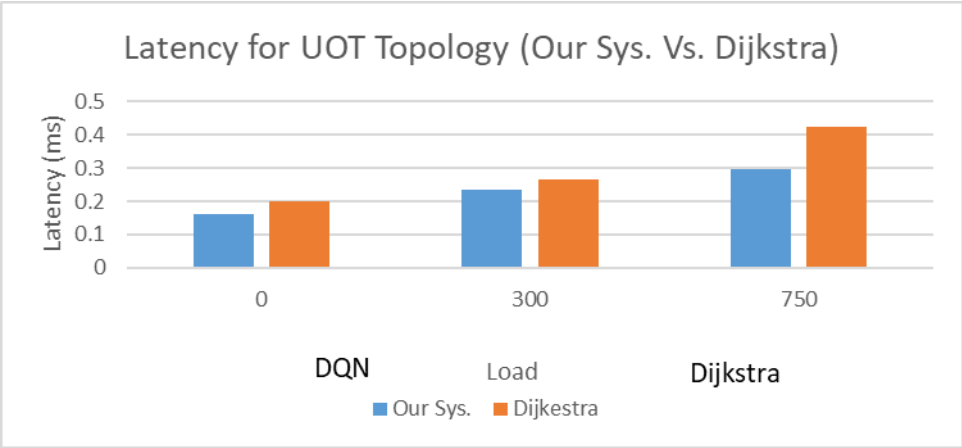


Fig. (4.19) Latency for UOT topology (Our Sys Vs. Adv Dijkstra)

The proposed system obtained an average “difference” of approximately −20.52% in latency at the three load levels

3. Figure 4.20 demonstrates, our proposed system enhances the jitter rather than the Advanced Dijkstra algorithm

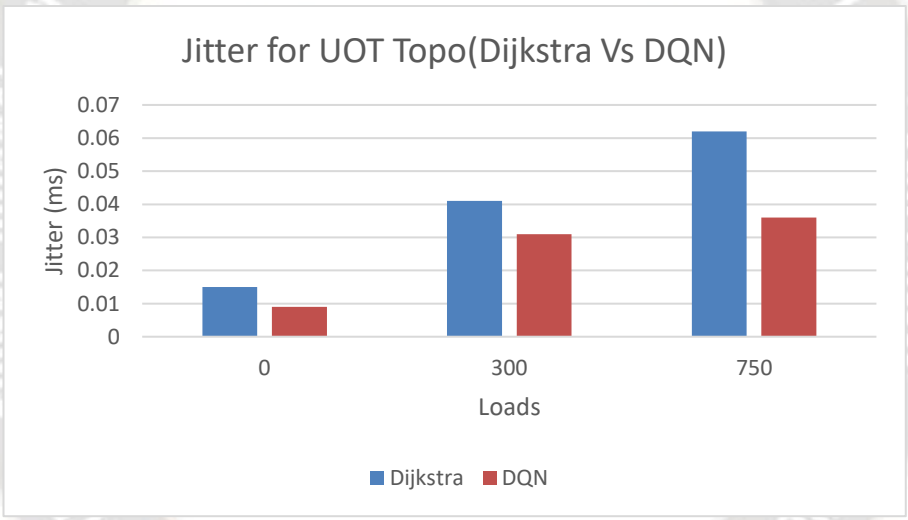


Fig (4.20) Jitter for UOT Topo (Our Sys. Vs. Adv Dijkestra

4.5.2) Experiment 2: Q learning algorithm and our proposed algorithm

A) Simple topology:

1. As shown in Figure 4.21, our proposed system increases the throughput rather than Q learning algorithm for all loads.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

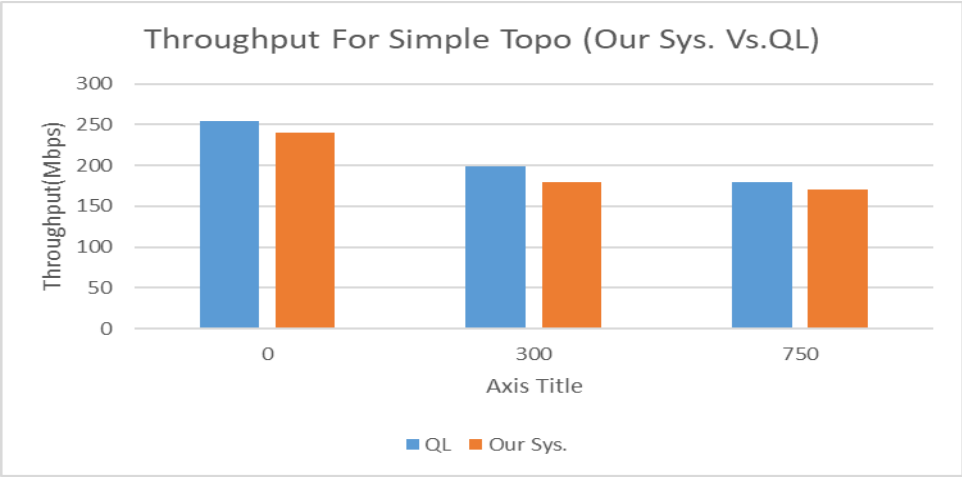


Fig. (4.21) Throughput for Simple Topo (Our Sys. Vs. QL)

The performance of the proposed system (approximately 10%, 6%, and 7% at the 0, 300, and 750 load levels, respectively) was lower than that of the QL

1. Figure 4.22 illustrates our proposed system has better performance in zero, 300, and 750 loads with less latency rather than Q learning. Furthermore, the average "difference" in latency of the QL was approximately −28.23% better than that of DQN at all load levels.

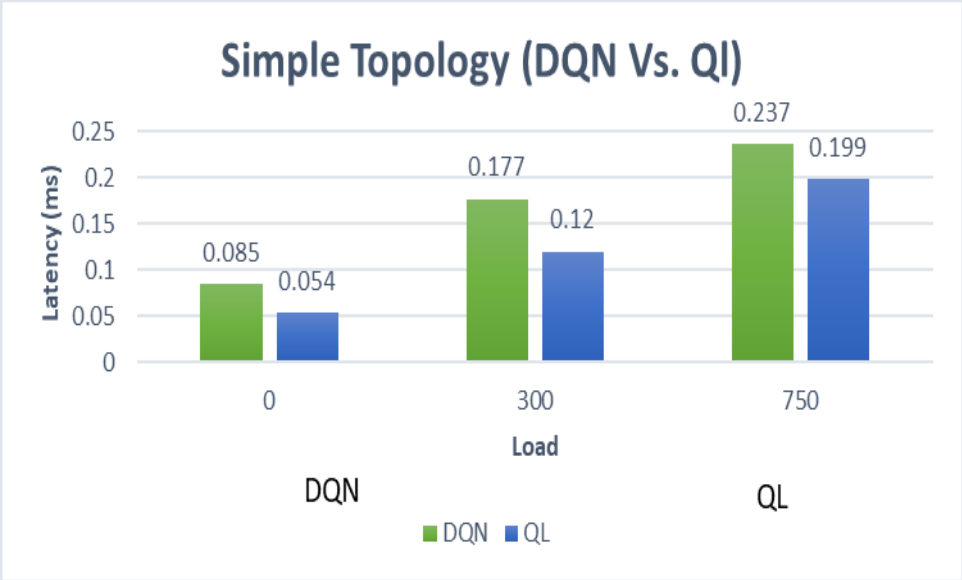


Fig (4.22) Latency for Simple topo (Our Sys. Vs. Q L)

3. As shown in Figure 4.23, our proposed system enhances the jitter rather than the Advanced Dijkstra algorithm

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: 25 August 2023 *Revised:* 05 October 2023 *Accepted:* 24 October 2023

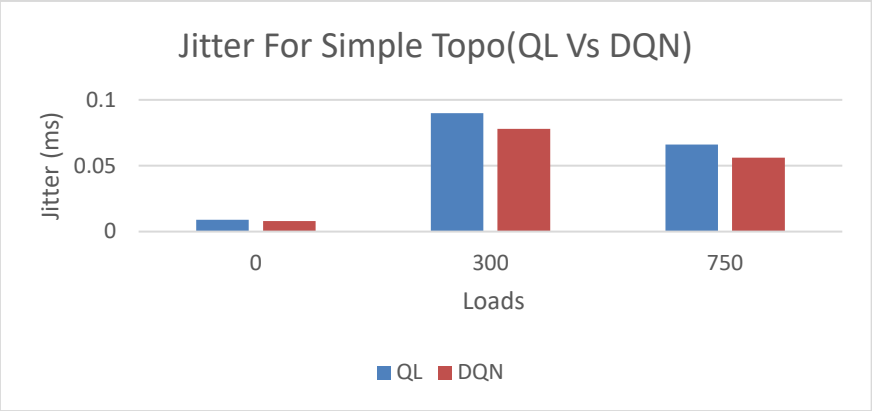


Fig (4.23) Jitter for Simple Topo (Our Sys. Vs. QL)

B) medium topology

1. Figure 4.24 demonstrates our proposed system increases the throughput rather than Q learning algorithm for all loads. The average "difference" in the throughput of DQN was approximately 20.49% higher than that of the QL at all load levels

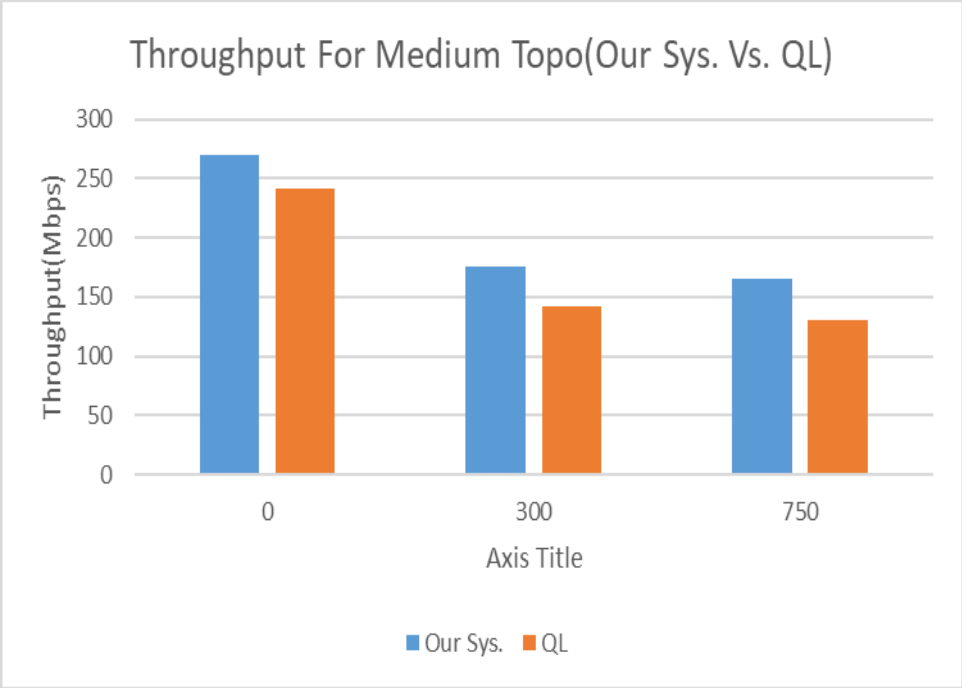


Fig. (4.24) Throughput for Medium Topo (Our Sys. Vs. QL)

2. As shown in Figure 4.25, our proposed system has also better performance in zero, 300 and 750 loads with less latency rather than Q learning. The average "difference" in latency of the DQN was approximately −10.01% with respect to that of QL at all load levels

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

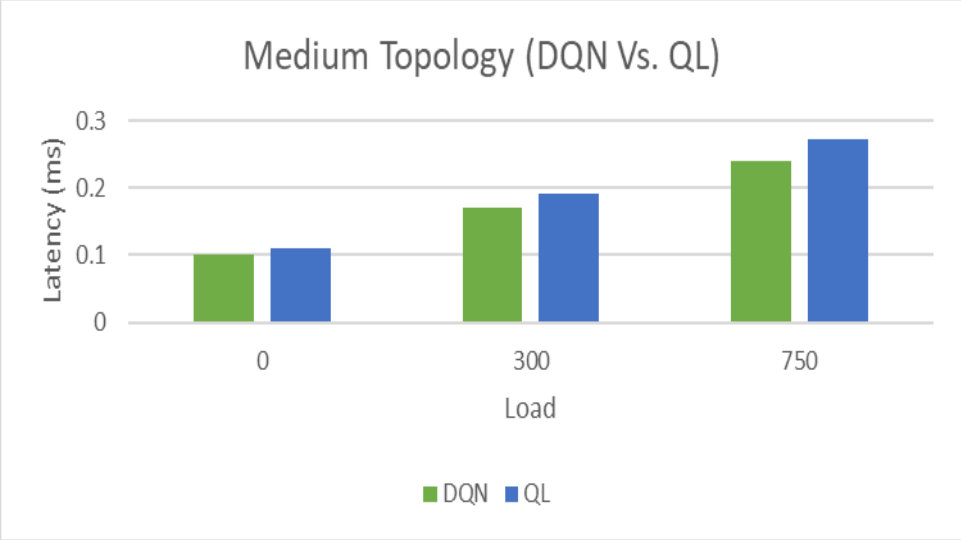


Fig. (4.25) Latency for medium topology (Our Sys. Vs. QL)

3. As shown in Figure 4.26, our proposed system enhances the jitter rather than the Q learning algorithm

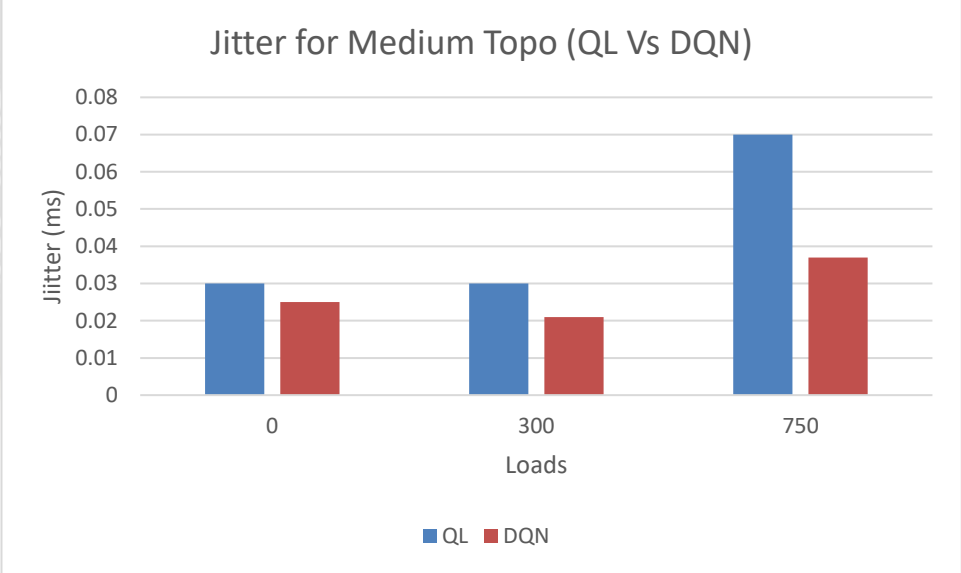


Fig (4.26) Jitter for Medium Topo (Our Sys. Vs. QL)

C) UOT topology

1. As shown in Figure 4.27, our proposed system increases the throughput rather than Q learning algorithm for all loads. Our proposed system shows an average “difference” of approximately 36.20% in throughput compared with QL at the three load levels

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

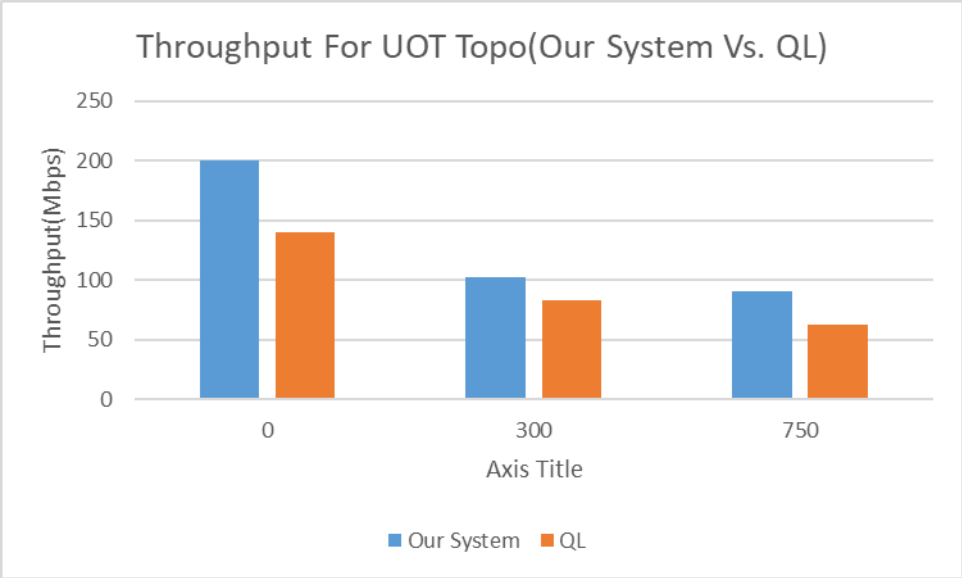


Fig. (4.27) Throughput for UOT Topo (Our Sys. Vs. QL)

2. Figure 4.28 demonstrates our proposed system has also a better performance in zero, 300 and 750 loads with less latency rather than Q learning. DQN shows an average "difference" of approximately −21.87% in latency compared with QL at the three load levels.

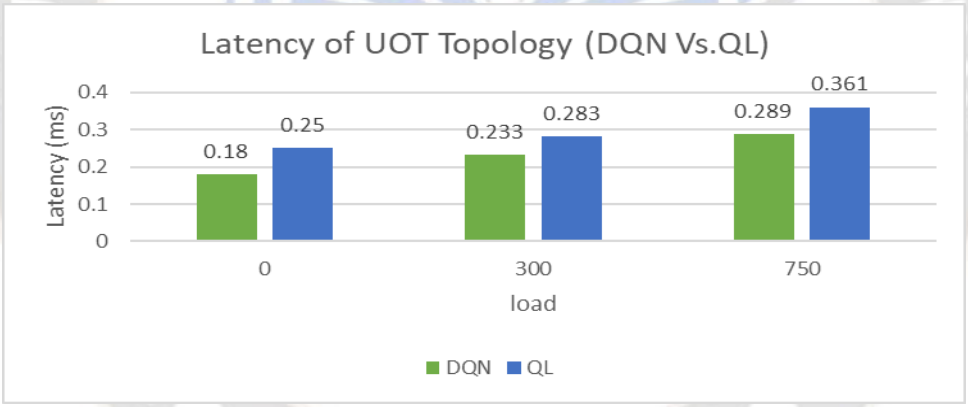


Fig. (4.28) Latency of UOT topology (Our Sys. Vs. QL)

3. As shown in Figure 4.29, our proposed system enhances the jitter rather than the Q learning algorithm,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 August 2023* ***Revised:*** *05 October 2023* ***Accepted:*** *24 October 2023*

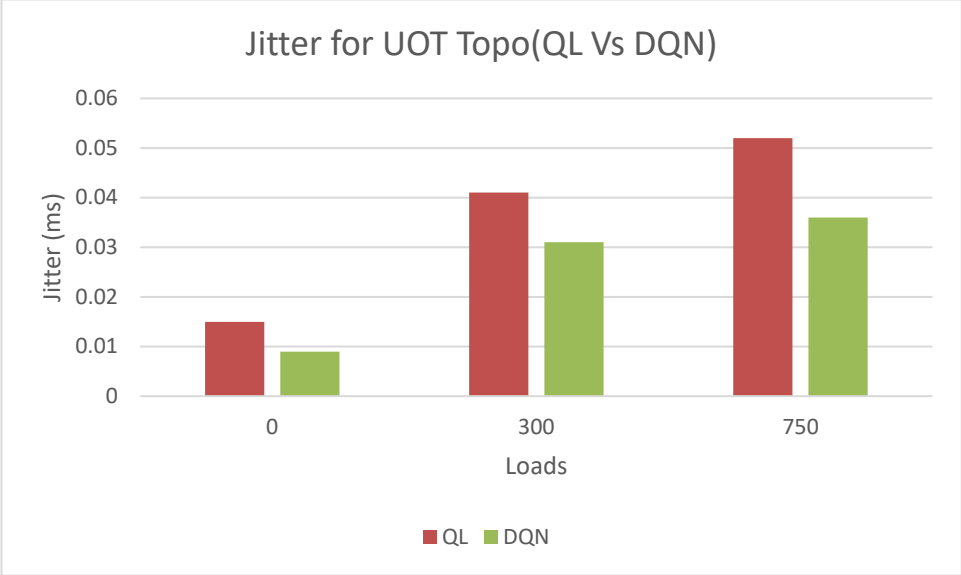


Fig (4.29) Jitter for UOT Topo (Our Sys. Vs. QL)

Conclusion and Future work

The campus network is a crucial part of university infrastructure, especially when it offers high-speed, reliable, and secure network access to students, faculty, and staff. A network should have the capability to handle many users regardless of access time. A network must also be able to manage the growing demands of new technologies, such as cloud computing, big data analytics, and IoT devices. Meeting these demands requires campus networks to be highly scalable, flexible, and adaptable, with the ability to add or remove network components as required. SDNs have emerged as a promising solution to the weak performance and security of campus networks. SDNs allow the campus network to be programmed and managed using a software instead of relying on the manual configuration of hardware devices. The relevant points and results of this research can be summarized as follows:

- The system proposed in this study applied the DRL algorithm in a campus network, and UOT was taken as the case study. The proposed system employed a two-method approach for optimizing network QoS. First, service types were classified, and TCP traffic was directed via DQN for intelligent routing. Then, UDP traffic was managed using the Dijkstra algorithm for shortest-path selection.
- The proposed system used available bandwidth and delay links as metrics and weighted them to compute the rewards of the environment.
- Designing and implementing the advanced Dijkstra algorithm also involved making a fair compression with the proposed system.
- The QL algorithm was implemented, and the obtained result was evaluated with respect to that of the proposed system.
- In addition to designing the network topology of OUT, two custom network topologies were designed and applied under three load conditions. The performance of each network topology was evaluated.
- The results demonstrate the relatively good performance of the proposed system, as evidenced by the increasing throughput and decreasing latency, for medium and complex topologies.

Future work:

- **Extension to other networks**. The proposed model can be extended to other types of networks, such as data centers or large-scale WANs. Their scalability and effectiveness in different environments must be evaluated.
- **Quality of experience (QoE)**: The present study, which focused solely on QoS, may be applied to the research of QoE of end-users.

Security concerns: Future work may focus on ensuring the security and privacy aspects of network systems.

References

[1] Puślecki, Ł., Challenges for innovation cooperation in the biopharmaceutical industry during the Covid-19 pandemic, in Toward the „new normal" after COVID-19–a post-transition economy perspective. 2021, Wydawnictwo Uniwersytetu Ekonomicznego w Poznaniu. p. 196-208.

[2] Triyason, T., A. Tassanaviboon, and P. Kanthamanon. Hybrid classroom: Designing for the new normal after COVID-19 pandemic. in Proceedings of the 11th International Conference on Advances in Information Technology. 2020.

[3] Ran, D. Design and Planning of University Campus Network. in Journal of Physics: Conference Series. 2020. IOP Publishing.

[4] Al-Sadi, A.M., et al. Developing an Asynchronous Technique to Evaluate the Performance of SDN HP Aruba switch and OVS. in Intelligent Computing: Proceedings of the 2018 Computing Conference, Volume 2. 2019. Springer.

[5] Rojas Sánchez, E., et al., Challenges and Solutions for hybrid SDN. 2021.

[6] Khin, C.S., et al., Reducing Packet-In Messages in OpenFlow Networks. ECTI Transactions on Electrical Engineering, Electronics, and Communications, 2022. **20**(1): p. 1-9.

[7] Li, D., et al., A survey of network update in SDN. Frontiers of computer science, 2017. **11**: p. 4-12.

[8] Ai, J., et al., Improving the routing security in software-defined networks. IEEE Communications Letters, 2019. **23(**5): p. 838-841.

[9] Nunes, B.A.A., et al., A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications surveys & tutorials, 2014. **16**(3): p. 1617-1634.

[10] Ranjan, P., et al., A survey of past present and future of software defined networking. Hal, Tokyo, Japan, 2014.

[11] Kobayashi, M., et al., Maturing of OpenFlow and software-defined networking through deployments. Computer Networks, 2014. **61**: p. 151-175.

[12] Xu, H., et al., Incremental deployment and throughput maximization routing for a hybrid SDN. IEEE/ACM Transactions on Networking, 2017. **25**(3): p. 1861-1875.

[13] Binlun, J.N., et al. Challenges and Direction of Hybrid SDN Migration in ISP networks. in 2018 IEEE International Conference on Electronics and Communication Engineering (ICECE). 2018. IEEE.

[14] Shin, M.-K., K.-H. Nam, and H.-J. Kim. Software-defined networking (SDN): A reference architecture and open APIs. in 2012 International Conference on ICT Convergence (ICTC). 2012. IEEE.

[15] Yu, C., et al., DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. IEEE Access, 2018. **6**: p. 64533-64539.

[16] Javadpour, A., et al., SCEMA: an SDN-oriented Cost-effective Edge-based MTD Approach. IEEE Transactions on Information Forensics and Security, 2022. **18**: p. 667-682.

[17] Al-Sadi, A.M., et al. Routing algorithm optimization for software defined network WAN. in 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA). 2016. IEEE.

[18] Mousa, A.K. and M.N. Abdullah, A Survey on Load Balancing, Routing, and Congestion in SDN. Engineering and Technology Journal, 2022. **40**(10): p. 1284-1294.

[19] Venkatasai, R., U. Prabu, and S. Ch. A Survey and Analysis of QoS-based Routing Techniques in Software-Defined Networks. in 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS). 2023. IEEE.

[20] J. Chavula, M. Densmore, and H. Suleman, "Using sdn and reinforcement learning for traffic engineering in ubuntunet alliance," International Conference on Advances in Computing and Communication Engineering (ICACCE), pp. 349–355, 2016.

[21] S. Kim, A. Son, and C. Hong, "Congestion prevention mechanism based on q-leaning for efficient routing in sdn," n international Conference on Information Networking (ICOIN), pp. 124–128, 2016.

[22] Y. Shilova, M. Kavalerov, and I. Bezukladnikov, "Full echo q-routing with adaptive learning rates: A reinforcement learning approach to network routing," IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), pp. 341–344, Feb. 2016.

[23] S. Kumar and R. Miikkulainen, "Dual reinforcement q-routing: An on-line adaptive routing algorithm," Smart Engineering Systems: Neural Networks, Fuzzy Logic, Data Mining, and Evolutionary Programming, vol. 7, 1997. [Online]. Available: http://nn.cs.utexas.edu/?kumar:annie97

[24] Sendra, S.; Rego, A.; Lloret, J.; Jimenez, J.M.; Romero, O. Including artificial intelligence in a routing protocol using software defined networks. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICCWorkshops), Paris, France, 21–23 May 2017; pp. 670–674.

[25]. Hossain, M.B.; Wei, J. Reinforcement learning-driven QoS-aware intelligent routing for software-defined networks. In Proceedings of the 2019 IEEE global conference on signal and information processing (GlobalSIP), Ottawa, ON, Canada, 11–14 November 2019; pp. 1–5.

[26] Rischke, J., Sossalla, P., Salah, H., Fitzek, F.H. and Reisslein, M., 2020. QR-SDN: Towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks. IEEE Access, 8, pp.174773-174791.