

**A Q-LEARNING BASED LOAD BALANCED AND
QOS-AWARE SDN APPROACH: A CASE STUDY IN
DEFENCE INDUSTRY**

**Q-ÖĞRENME TABANLI YÜK DENGELEME VE SERVİS
KALİTESİ FARKINDA BİR YTA YAKLAŞIMI: SAVUNMA
SANAYİİ ENDÜSTRİSİNDE BİR VAKA ÇALIŞMASI**

TEVFİK AKTAY

ASSOC. PROF. DR. CEREN TUNCER ŞAKAR

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Industrial Engineering

May 2022

ABSTRACT

A Q-LEARNING BASED LOAD BALANCED AND QOS-AWARE SDN APPROACH: A CASE STUDY IN DEFENCE INDUSTRY

Tevfik Aktay

Master of Science , Industrial Engineering

Supervisor: Assoc. Prof. Dr. Ceren TUNCER ŞAKAR

May 2022, 108 pages

Traditional network routing methods are insufficient in the face of exponentially increasing data, device diversity and service demands' variety, making the necessity of more manageable routing methods felt at both the Internet and Local Area Networks (LAN). It is predicted that the Software Defined Networks (SDN) are going to be able to manage the data traffic of the digital world in a more democratic way with smart algorithms, due to their programmable and centralized management structure. In this study, the network environment of a defense industry company that has a LAN infrastructure on a large campus was selected as a case study. This internal network serves a a wide variety of devices with dozens of switches. A part of its topology and its traffic that currently routed with Spanning Tree Protocol (STP) has been simulated with widely used methods. In this network environment; various test scenarios have been studied with STP, a rule-based SDN, and our Q-learning based SDN approach. Our proposal, which performs the load balancing of the system while providing the requested QoS to the clients, has achieved effective results under various QoS performance metrics and load balancing indicators. **Keywords:** software defined networking, load balancing, quality of service, reinforcement learning, machine learning, Internet of Things

ÖZET

Q-ÖĞRENME TABANLI YÜK DENGELEME VE SERVİS KALİTESİ FARKINDA BİR YTA YAKLAŞIMI: SAVUNMA SANAYİİ ENDÜSTRİSİNDE BİR VAKA ÇALIŞMASI

Tevfik Aktay

Yüksek Lisans, Endüstri Mühendisliği

Danışman: Doç. Dr. Ceren TUNCER ŞAKAR

Mayıs 2022, 108 sayfa

Geleneksel ağ rotalama yöntemleri üstel artan veri, cihaz çeşitliliği ve değişken hizmet talepleri karşısında yetersiz kalmakta ve daha yönetilebilir rotalama yöntemlerinin gerekliliğini hem Internet hem de LAN (İng. Local Area Networks) seviyesinde hissettirmektedir. SDN (İng. Software Defined Networks) yaklaşımının programlanabilir ve merkezi yönetim yapısı sayesinde dijital dünyanın veri trafiğinin akıllı algoritmalar ile daha demokratik şekilde yönetebileceği öngörülmektedir. Bu çalışmada, LAN altyapısına bir savunma sanayi şirketinin ağ ortamı vaka çalışması olarak belirlenmiştir. Bu iç ağ, onlarca anahtarlayıcı ile çok çeşitli istemcilere hizmet vermektedir. Mevcutta STP (İng. Spanning Tree Protocol) ile yönlendirme yapan tesisin ağ topolojisinin bir bölümü ve trafiği kabul gören yöntemler ile simule edilmiştir. Bu ağ ortamında; STP, kural tabanlı bir SDN yaklaşımı ve Q-öğrenme tabanlı SDN yaklaşımımız ile çeşitli test senaryoları çalışılmıştır. İstemcilere talep ettiği QoS (İng. Quality of Service) sağlarken sistemin yük dengelemesi de gerçekleştiren önerimiz çeşitli QoS performans metrikleri ve yük dengeleme göstergeleri altında etkin sonuçlar elde etmiştir. **Anahtar Kelimeler:** Yazılım Tanımlı Ağlar, Servis Kalitesi, Yük Dengeleme, Q-Öğrenme, Makine Öğrenmesi, Nesnelerin İnterneti

ACKNOWLEDGEMENTS

I would like to present my greatest appreciation to Assoc. Prof. Dr. Oumout Chouseinoglou for his support that gave me confidence in the selection of this thesis topic and its continuation. Since my undergraduate years at Hacettepe University, he gave valuable advices to me when I needed it. His influence is important in my development in computer science, which I am deeply interested in. I would also like to thank Dr. Cüneyt Sevgi for encouraging me in determining the subject of this thesis. They gave me challenging topics to work on and their questions provide me the opportunity to think about many different angles in this thesis. Additionally, I would like to express my gratitude to my supervisor Assoc. Prof. Ceren Tuncer Şakar. I hope that I learned a lot from her in terms of timing and coordination.

I also would like to express my gratitude to the valuable members of the jury, Prof. Dr. Mehmet Önder Efe, Prof. Dr. Özlem Müge Testik, Assoc. Prof. Barbaros Yet and Asst. Prof. Erdi Daşdemir, who took part in the examination of my thesis and give helpful feedbacks.

I would like to thank my friends Mehmet Emin Gülşen, Ahmet Serdar Karadeniz and Mehmet Fatih Karadeniz, Mustafa Çağrı Güven who have been with me since the beginning of my thesis journey. We have done good works in a start-up and will continue to do so. I would also like to thank my friend Oğul Can Eryüksel, who has always inspired me with his approach to technical issues.

I would like to express my biggest thanks to my family. They have always supported me unconditionally and I have always felt that. Special thanks to my dear mother, her support was priceless and I dedicate this work to her.

As a final word, I would like to thank my institution, Turkish Aerospace, which supports my work and provides the technical infrastructure I used in this study.

CONTENTS

	<u>Page</u>
ABSTRACT	i
ÖZET	ii
ACKNOWLEDGEMENTS	iii
CONTENTS	iv
TABLES	vii
FIGURES	viii
ABBREVIATIONS.....	x
1. INTRODUCTION	1
1.1. Scope of the Thesis	4
1.2. Contributions	5
1.3. Methodology of the Thesis.....	6
1.4. Organization	7
2. BACKGROUND OVERVIEW	8
2.1. Q-Learning.....	8
2.2. Traditional TCP/IP Networks	9
2.3. Spanning Tree Protocol	11
2.4. OpenFlow	11
2.5. Software Defined Networking	13
2.5.1. Components of SDN	16
2.5.1.1. Data Layer	17
2.5.1.2. Control Layer	17
2.5.1.3. Application Layer	18
2.6. Quality of Services	18
2.7. Mininet.....	19
2.8. Ryu Controller	20
3. RELATED WORK.....	22
3.1. Single-factor centered QoS approaches	24

3.2. Multi-factor centered QoS approaches	25
4. PROPOSED METHOD.....	28
4.1. Network Traffic and Route Awareness Module	29
4.1.1. Route Discoverer	29
4.1.2. Network Statistics Tracker.....	30
4.2. QoS and Load Awareness Module	31
4.2.1. Available Bandwidth Tracker.....	32
4.2.2. Delay Time Tracker	33
4.2.3. Route Hop Tracker	36
4.2.4. Switch Load Tracker	39
4.3. Route Selector Module	39
4.3.1. Equally QoS-aware Route Selector	42
4.3.2. Q-learning based Adaptive QoS and Load-aware Route Selector	43
4.3.2.1. Overview	43
4.3.2.2. Environment	44
4.3.2.3. State	44
4.3.2.4. Action	46
4.3.2.5. Reward	47
4.3.2.6. The Optimal Policy	48
4.3.2.7. Exploration and Exploitation Technique	50
4.3.2.8. Training the Q-learning model	50
4.4. Flow Starter Module.....	52
5. EXPERIMENTS AND RESULTS	53
5.1. Test Environment	53
5.2. Flow Generation	54
5.3. Test Scenarios.....	55
5.4. Performance Criteria	56
5.5. Tuning the proposed RL-based Approach.....	57
5.6. Results and Analysis	59
5.6.1. Bandwidth provisioning capacity	60

5.6.2. Average delay time.....	61
5.6.3. Average jitter time	62
5.6.4. Total packet loss.....	62
5.6.5. Average received packet per second.....	63
5.6.6. Flow duration.....	64
5.6.7. System load balance	67
6. CONCLUSION	71
Appendix	87

TABLES

	<u>Page</u>
Table 3.1 QoS factor optimization centered researches with details.....	23
Table 4.1 Data types that collected periodically from each port on switches	31
Table 4.2 Equal predefined QoS factor weight values.....	43
Table 4.3 Conversion table of continuous load percentage to the discrete position for switches	45
Table 4.4 Action numbers and their weight equivalents in discrete range.....	47
Table 5.1 Predefined flow size and rate sets for mice and elephant flows	55
Table 5.2 Suggested traffic scenarios for tests	56
Table 5.3 Statistical evaluation of loads on the time intervals for each routing methods	69

FIGURES

		<u>Page</u>
Figure 1.1	Global device and connection growth [1]	1
Figure 2.1	Typical framing of the RL scenario [2]	8
Figure 2.2	System architecture of OpenFlow protocol which is adopted from official document [3]	13
Figure 2.3	An overview of the 3-layered SDN architecture which is adopted from [4]	15
Figure 2.4	Overview of the SDN architecture with Ryu Controller [5]	21
Figure 4.1	Flow diagram of periodical bandwidth tracking process	33
Figure 4.2	Flow diagram of available bandwidth determination process of a route	34
Figure 4.3	Flow diagram of periodical delay time tracking process	35
Figure 4.4	An illustration of the delay measurement procedure in a mini topology	36
Figure 4.5	Flow diagram of total delay time determination process of a route.....	37
Figure 4.6	Flow diagram of total hop count determination process of a route	38
Figure 4.7	Comparative overview of the proposed RL-based controller (Design A) and rule-based controller (Design B) frameworks	42
Figure 4.8	Proposed Q-learning based routing approach on computer networks domain (adopted from [2], and modified in networking domain).....	44
Figure 4.9	An illustration of mesh-hierarchical topology definition as Data Layer with OpenFlow messages (taken from [6])	45
Figure 4.10	Q-table visualization that shows Q-values by index of actions and states	49
Figure 5.1	An illustration of the hierarchical and mesh topology design for the simulations	53
Figure 5.2	EMA of R for different lr , $\gamma = 0.1$	59
Figure 5.3	EMA of R for different lr , $\gamma = 0.3$	59
Figure 5.4	EMA of R for different lr , $\gamma = 0.6$	59
Figure 5.5	EMA of R for different lr , $\gamma = 0.9$	59

Figure 5.6	Bandwidth provisioning capacities of the routing approaches	60
Figure 5.7	Average delay time results of routing algorithms according to scenarios	61
Figure 5.8	Average jitter time results of routing algorithms according to scenarios	62
Figure 5.9	Total packet loss results of routing algorithms according to scenarios .	63
Figure 5.10	Average received packet per second results of routing algorithms according to scenarios	64
Figure 5.11	Flow duration analysis for Scenario 1	65
Figure 5.12	Flow duration analysis for Scenario 2.....	66
Figure 5.13	Flow duration analysis for Scenario 3.....	66
Figure 5.14	Cumulative received data amount on core and distribution switches for our proposed routing method	67
Figure 5.15	Cumulative received data amount on core and distribution switches for equally weighted QoS routing method	68
Figure 5.16	Cumulative received data amount on core and distribution switches for STP routing method	69
Figure 6.1	Topology initialization process using Mininet.....	87
Figure 6.2	Scenario initializations. Predefined flow settings and traffic details is shown	88
Figure 6.3	Flow initialization command on receiver host using <i>iperf</i> tool	89
Figure 6.4	Flow initialization command on sender host using <i>iperf</i> tool.....	90
Figure 6.5	Periodic data transfer report that is written in receiver host using <i>iperf</i> tool.....	91
Figure 6.6	CMA of R for different lr , $\gamma = 0.1$	92
Figure 6.7	CMA of R for different lr , $\gamma = 0.3$	92
Figure 6.8	CMA of R for different lr , $\gamma = 0.6$	92
Figure 6.9	CMA of R for different lr , $\gamma = 0.9$	92

ABBREVIATIONS

API	: Application-Programming-Interface
ATM	: Asynchronous-Transfer-Mode
BFS	: Breadth-First-Search
CAN	: Campus Area Networks
DFS	: Depth-First-Search
IoT	: Internet-of-Things
IP	: Internet Protocol
LAN	: Local Area Network
LLDP	: Link Layer Discovery Protocol
ML	: Machine Learning
ONF	: Open Networking Foundation
QoE	: Quality-of-Experience
QoS	: Quality-of-Service
OSPF	: Open-Shortest-Path-First
RIP	: Routing Information Protocol
RL	: Reinforcement Learning
SDN	: Software Defined Networking
STP	: Spanning Tree Protocol
SVM	: Support Vector Machines
TCP	: Transmission Control Protocol
UDP	: User Datagram Protocol
WAN	: Wide Area Networks

1. INTRODUCTION

Routing of the messages and data between any internet users or devices from source to destination is the most crucial role of the computer networks. In addition to the continuity and security of this task, Quality of Service-aware (QoS) message transfer is necessary for the appropriate service experience. For a long time, Open Shortest Path Protocol (OSPF) and Routing Information Protocol (RIP) have been preferred on Internet scale and also Spanning Tree Protocol-like (STP) protocols are used on local scale networking to realize best-effort based routing solutions. However, the number of connected device to the networking infrastructure globally is increasing day by day due to the cheaper chip technology and digitization trends. Cloud-based services and Internet of Things (IoT) products with the power of 5G connection technology accelerates this technology transition. According to the recent annual report [1], There will be 3.6 network-connected devices per capita by 2023. Additionally, there will be 29.3 billion networked devices by 2023, up from 18.4 billion in 2018. Machine-To-Machine (M2M) connections will be half of the global connected devices and connections by 2023. The share of M2M connections will grow from 33 percent in 2018 to 50 percent by 2023. Therefore, it poses a serious challenge to overcome the large number of connections created by a large number of M2M devices. Best effort procedural routing methods have fundamental limitations that can cause undue congestion and chaos.

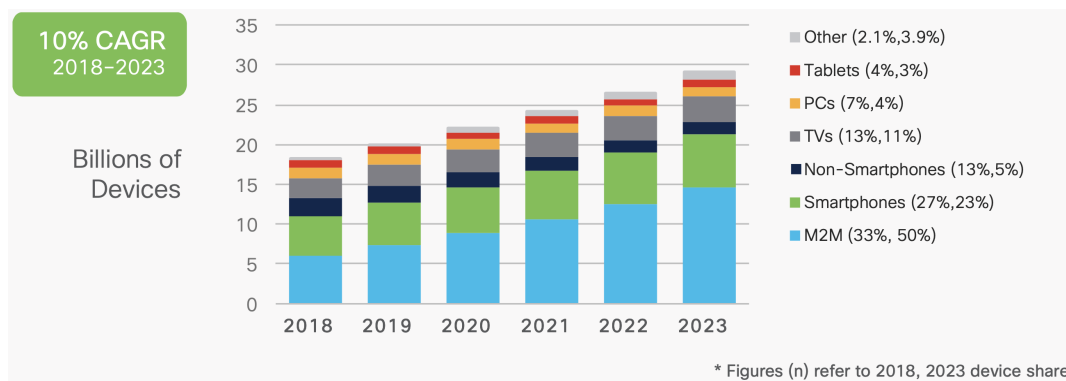


Figure 1.1 Global device and connection growth [1]

As shown in Figure 1.1, many network-connected device types as PCs, smartphones and IoT-based M2M devices increases year by year. Additionally, by 2025, the estimated data

amount is 463 exabytes for each day globally [7]. The increasing speed on number of connected devices and their data amount is higher than the improvement works in networking infrastructure. Also, QoS requirements (delay, jitter, loss and throughput rate, bandwidth) of the flow requests are vary according to the application type. For instance, jitter and latency may be very crucial for users during a video conference, data loss rate may be very important for an application that run on a server which control hardware health of a reactor by performing time series analysis. For this reason, best effort-based routing protocols fails day by day due to increase of new traffic patterns and flow requirements. At this point, the communication capability to the switches via OpenFlow [8] protocol and the programmability to the edge network devices with Software Defined Networks (SDN) [3] are revolutionary in computer networking. Taking the routing decision from the edge network devices and leaving only the forwarding task makes it possible to configure the network with cheaper and simpler devices, and also allows network administrators to manage the entire system from a central point. In this way, sustainable management of the system is easier under the increased demand for the networking requests from users, applications or any network-connected devices. Due to these possibilities, the SDN approach is included in various studies of companies such as Google [9].

There are many studies which are trying to enhance the required QoS and quality of experience (QoE) factors for the end user or applications in years. These works begin in conventional Internet era [10] and continue with the OpenFlow and SDN paradigms [11]. On the other hand, various load balancing approaches for optimal management of the network systems are exist while meeting the demands of the flows that constitute the traffic [12]. Some of these approaches are proposed as deterministic and rule-based frameworks as well as also machine learning-based (ML) stochastic approaches are preferred to learn the unstable and dynamic network traffic environment. In this context, our research questions are listed below:

- RQ-1: Can QoS and load balancing subjects which are often handled separately, be addressed simultaneously?

- RQ-2: Is it possible to utilize the capacity of the network in a balanced way and to meet the QoS demand of the flows equally and democratically?

The only enhancement effort for the optimizing and balancing system resources may cause in QoS problems for flows. In another way, only flow-oriented QoS efforts can result unbalanced system usage. Our research is centered to solve this problem.

Today, Campus Area Networks (CAN) which are a preferred network type for reasons such as security and privacy. Especially, manufacturing facilities and research centers prefer them due to the necessary regulations. Defence companies which are one of the stakeholder for CAN systems, utilizes on premise networking architecture to connect their shop floor machines and personal work stations. They connect manufacturing devices, workstations, mobile robots, IoT sensors, and personal devices to interior networks and that enables end-to-end monitoring and management of the production. Also, solutions are developed over the networks for planning, maintenance, breakdown and bottleneck situations. These networks can be configured with conventional networking paradigms like the ones used for Internet, as well as SDN-like new generation networking paradigms can be preferred.

Thanks to the disruptive effect of Industry 4.0, IoT and artificial intelligence, it is foreseen a transition from traditional networking to SDN on defence company campuses. The networks which is deployed on these manufacturing areas requires a comprehensive QoS management and supply strategy for network clients. Many applications and services uses a common infrastructure with limited resources. While the sensor data obtained from the manufacturing robots have to send it to the data center with the least loss rates, it is critical to transmit the test data obtained from the end products to the servers with the minimum delay and jitter. In addition, the fact that many devices transfer data with each other without a certain rule can cause congestion, data loss and delays that are difficult to predict by network administrators. Such unstable situations can lead to disruptions in critical processes. Therefore, an approach that both monitors and balances the overall state of the networks and provides the best QoS regardless of flow type is essential for the sustainability of the production facility.

In this thesis, we simulated a part of interior networks which is configured as conventional networking on Turkish Aerospace (TA). The constraints and difficulties around QoS and load balancing in the existing IP network in TA campus are revealed. In order to eliminate these disadvantages, using the SDN approach in an area on the defence industry campus is recommended as a case study. In various traffic scenarios, we propose an adaptive and intelligent SDN-based approach to solve problems that are arising under the traditional networking.

1.1. Scope of the Thesis

This thesis mainly focuses on providing a load-balancing and QoS aware routing framework with SDN for CAN. The proposed method is partially local but has generalizable flexibility, thus it can be applied to more expanded networks systematically. In networking research, QoS is a significant context that consist of latency, jitter, packet loss rate and throughput rate which are crucial for a healthy networking experience for end users or host applications. Our focused network environment in this research has a harmonic traffic which requests from ordinary end user devices as on Internet, and also IoT-based flows from sensors. Therefore, best effort routing for various applications is insufficient for QoS optimization and a balanced network utilization. For this purpose, the proposed method in this study is based on the SDN approach in terms of instant monitoring and programmability. In this way, the routing capability separated from the edge networking devices, and central controller structure is established.

The density of the traffic varies and is unstable in networking. Additionally, flow variability and their service requirements makes it difficult to design deterministic system design with SDN. As proposed in previous works [13–15], classifying flows as mice and elephant and routing with protocols causes some neglections and insufficient QoS allocation for unvalued flows. On the other hand, routing according to the least congested path or flow aggregation techniques [16, 17] while performing load balancing of the system may result in inefficient use of the system resources. For this reason, effective management of the network

environment is a stochastic problem, and it is appropriate to manage it by considering both the QoS for flows and the system side for network resources. In this context, we proposed a reinforcement learning (RL) based approach that cares for load balancing of the system resources when meets optimal QoS needs for each flow regardless of its classes. Firstly in training phase, our method learns the traffic patterns and system overview (that is called environment). In running phase, when a new flow request comes in to the controller, it analyses the performance metrics that are generated by traffic (this is called the state, and in this study, it is the available bandwidth, delay time and total number of hops of the appropriate routes that links source and destination nodes). Then, controller selects the most scored route according to the actual system load state (this is action, the RL agent recommends the most ideal weights for performance metrics according to the current traffic of the system, and the route with the highest QoS score is selected). Next, the network environment feedbacks the controller as reward or penalty for the new traffic status after the last routing decision (the reward function is designed specific to the problem in each system. In this study, the more evenly the load distribution on the data layer switches is distributed as a reward, and if it is unbalanced distributed, it is reflected back as a penalty). In this way, while routing each flow request, it is provided with a QoS optimized for delay, hop count and bandwidth as well as load balancing is performed on the switches to prevent unbalanced congestion and data loss. At the end, the proposed adaptive routing method has been tested with an SDN approach that equally weighted QoS factors and STP, a traditional routing method. With this intention, it is aimed to emphasize the importance of intelligent routing methods in stochastic network environments.

1.2. Contributions

In this thesis, we cover the aforementioned deficiencies by proposing a novel, simple and efficient approach. The main contributions can be summarized as follows:

- We propose a customizable routing framework that is designed as a prototype for CAN type networks. The framework is scalable for various size of networks. On a different

topology and QoS requirements, network administrators or researchers can define their constraints and objective functions. After that, retraining the new setup is needed to perform customized controller.

- Unlike most of the previous works, the proposed routing method makes the democratization of the load passing through the switches on the network, and has a QoS supply strategy that takes into account the optimal number of hops, delay time and bandwidth requirements for each flow are provided.
- With the proposed approach, the power of the best-effort routing method is added to our adaptive routing method by including the number of hops in the routing decision.
- As an intelligent algorithm choice that does not require much resources, Q-learning and a reward function design that can be customized by system administrators according to the environment in which it is applied are presented.
- Our simulation test results show that our method is relatively more efficient both in terms of load distribution on the switches and meeting the QoS requirements than an equally weighted rule-based SDN approach and the traditional STP routing approach.

1.3. Methodology of the Thesis

While preparing this thesis, basically the following method was followed. In the first stage, a literature review was studied on SDN for problem identification. After finding the problem and determining the our research questions, the company topology for the network topology to be taken as a basis for this study and transferred to the simulation environment. STP, which is the routing protocol of the existing network, has been designed appropriately. Additionally, one rule-based SDN and the other is Q-learning based SDN routing approaches have been proposed as routing methods. Each routing method was run on three different traffic with different characteristics and their logs were recorded. In order to test the performance of routing approaches within the scope of QoS and load balancing, seven different test methods were created and the test results were evaluated.

1.4. Organization

The organization of the thesis is as follows:

- Chapter 1 presents our motivation, contributions and the scope of the thesis.
- Chapter 2 provides brief explanations on the topics and technologies covered in the thesis.
- Chapter 3 gives previous related studies to the field of the proposed study. It is aimed to give detailed information in terms of both the problems and solution methods. In addition, the difference of the proposed method in this research from the existing studies is expressed.
- Chapter 4 introduces the whole details of the proposed routing method.
- Chapter 5 demonstrates the experimental design and test scenarios. In addition, the results of these experiments are evaluated.
- Chapter 6 states the summary of the thesis. Also, we provide a critics for proposed method and previous studies. Lastly, possible future directions for our work are presented.

2. BACKGROUND OVERVIEW

In this section, we provide descriptive information about the concepts we discussed in the Introduction section and technologies which are used in this study.

2.1. Q-Learning

RL is one of the three fundamental ML paradigms, the others are supervised learning and unsupervised learning. This approach deals with how intelligent agents should act in an environment in order to maximize the cumulative amount of reward [2].

Figure 2.1 represents the RL approach in which the RL agent observes the environment through the state space, while the reward value acts as a feedback mechanism for the agent. The agent transitions from state to state by taking actions. Next, the agent receives a reward for the action taken in a given state. The main goal of the agent is to maximize its reward. The potential reward is the weighted sum of the expected returns of future rewards from the current state.

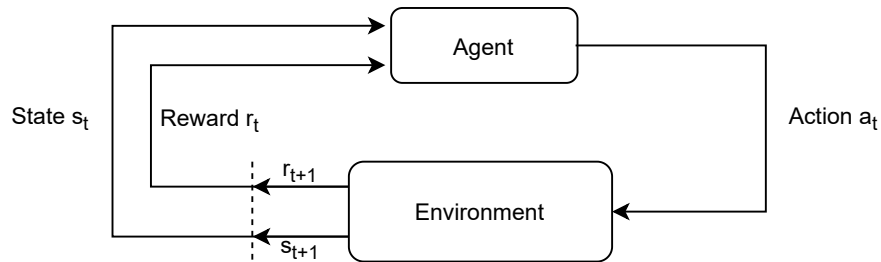


Figure 2.1 Typical framing of the RL scenario [2]

Q-learning is a model-free RL algorithm approach that learns the optimal policy or the quality of actions from a state over time [18]. It is a finite Markov Decision Process and its iterative approach tries to maximize the expected Q-value of the total reward for all the states, starting from the current state. At the beginning, states and actions have to be defined explicitly. Also, values in Q-table are initialized with zeros and number of states and actions are preferred to define number of rows and columns respectively. After a time step,

the agent chooses an action. Then, it causes a new state and gets a reward. After that, agent updates the Q-table according to the Q-function. Through the trials, the agent learns to choose the best action for the current state and stores Q-values on the Q-table as state-action pairs.

2.2. Traditional TCP/IP Networks

In one of the conventional computer networks, TCP/IP, a five-layer Internet protocol stack design is defined. Each layer communicates with its neighbors and has its own tasks to sustain networking operations [19]. On the top, the application layer is placed and its elements are end-user applications such as a web browser, online video game or and e-mail client. Application layer packs the messages to send to the other host or hosts. In the fourth layer, the transport layer takes the messages. As the main task, the transport layer controls the order of the packets which are transferred to the other side. At this point, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) define the rule of thumb of the transmission according to the requirements. The third layer is the network layer and routers work in this layer to manage the two critical tasks in networking operation, routing and forwarding. This layer uses Internet Protocol (IP) and the datagram packets are forwarded according to their unique numbers. In the next layer, the data link layer connects to the nodes in the same medium. The last layer is the physical layer which is commissioned to transfer the packets between nodes in medium.

In traditional networking, each layer communicates only with its peers in the network. Also, each layer takes service from lower layers and gives service to upper layers. This situation requires an encapsulated messaging approach which consists of data and transferred packet to transfer appropriately. Transferred packet includes all required information for the destination host. Since this information packet is processed and turned into the routing actions separately on each router in the current network approach. In this situation, the management of the network is distributed over each device and is far from a central control. The distributed management and operational interdependency make it difficult to respond to

the problems that may occur in the network environment. Actual TCP/IP paradigm even provides a little mechanism to automatically respond for the issues. Network providers on large scale or network administrators at the local level need some management tools to navigate and manipulate network operations to respond to maintenance or fault fix tasks. Below, the most known problems and constraints encountered in traditional networks are mentioned:

- **Hardware dependencies and constraints:** To implement a high-level rule to the network, some prior low level configurations are implemented to all data layer devices. This is very likely to cause errors and complications, because each device and software can be used according to its own commercial infrastructure. Also, the network service providers desire to manage and maintain the whole topology for a long-term. However, due to commercial benefits, various financial obligations for the maintenance and updates of the devices cannot be avoided.
- **Lack of programmability:** Because of the distributed structure of the management in traditional networks, traffic engineers and network administrators have difficulties with programming like rule-based routing, traffic prioritization and load balancing for network elements.
- **Non-separation of operational roles:** The integrated structure for routing and forwarding on the same layer blocks the modularity. This makes difficult to route any flow request from the desired path.
- **Dependency on too many task-device peers:** There are additional network assets in the structure which are firewalls, repeaters, bridges. Adding a new device for each new mission adds another level of complexity over the existing build. According to the research, the number of devices on the internet is equal to the number of these indirect devices [20].

Due to the limitations and permanent problems which are mentioned above for conventional networks, it has prompted network researchers and engineers to work on new structures

to make them more manageable. A complete architectural change seems impossible. Additionally, replacing all network devices or integrating new devices immediately will incur additional costs to network owners. In this context, the SDN paradigm seems to be one of the options for such a transition.

2.3. Spanning Tree Protocol

STP which was invented by Radia Perlman, tries to prevent getting stuck in an infinite loop in the network [21]. The protocol is a Layer 2 network protocol used to prevent problems that arise when computers compete to use shared network paths on a local area networks (LAN). When too many computers try to send data at the same time, it affects overall network performance. Delay, jitter, congestion problems arise and these bring all traffic to a near halt. STP prevents the condition known as bridge looping. Bridge looping is when there are multiple links between two hosts, and messages are sent out of every point continually flooding the network. To reduce the likelihood of looping, STP divides a LAN into two or more network segments with a device called a bridge connecting any two segments. Each message goes through the bridge before being sent to the target node. The bridge determines whether the message is for a destination within the same segment. Network segmentation reduces competition for network path used by half and significantly reduces the chances of a network coming to a halt. STP is easy to use, proven to be effective, offers wide support for bridges and switches, provides link redundancy and prevents looping, and offers various backups in case the main connection falters. But with the increasing usage of virtualization technology in data centers, STP can not be able to handle increased input/output demands. And full network capacity is not realized when using STP because it restricts traffic.

2.4. OpenFlow

OpenFlow is a communication protocol that allows a server or a controller to tell network switches where to send packets. Its development began at Stanford University in 2008 and by

December 2009 Version 1.0 of the OpenFlow key specification was released [8]. OpenFlow is currently managed by the Open Networking Foundation (ONF) [22].

In a traditional network approach, each switch has its own software. Routing decision and packet forwarding happen on the same network device. With OpenFlow, packet transport decisions are centralized thus the network can be programmed independently of individual switches. An OpenFlow switch separates the routing and forwarding decisions. The forwarding part is located in the data layer and separated controller makes high-level routing decisions. Switches and controller communicate via the OpenFlow protocol. The combination of the OpenFlow and SDN paradigms allow for more efficient use of network resources than is possible with traditional networks. Using this interface, SDN controller alters the switch/router flow tables for routing decisions and also data layer devices push up current traffic metrics to controller for to the data-oriented routing actions. With OpenFlow, many features are quickly available for network administrators:

- fast deployment of network applications
- possibility to implement a regional network policy
- performance tracking and central optimization
- Instant metric tracking of data layer elements

OpenFlow connects properly with OpenFlow-enabled switches which maintain their flow tables according to the set of rules that are defined by the controller. Figure 2.2 demonstrates communication types between controller and data layer using OpenFlow.

If any incoming packet's destination is not matched on flow tables in the switch, packet is forwarded to the controller via OpenFlow. The controller specifies the high-level actions for the flow and sends the answer via OpenFlow protocol. On another side, data layer devices send current port and flow statistics to the controller.

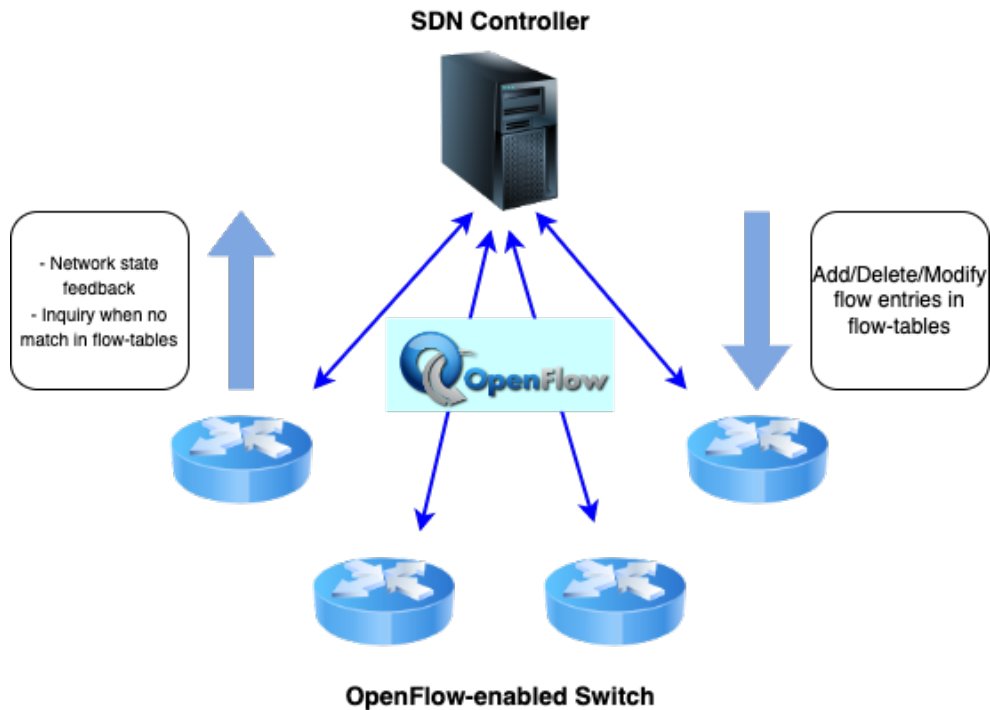


Figure 2.2 System architecture of OpenFlow protocol which is adopted from official document [3]

2.5. Software Defined Networking

The efforts to make computer networks more flexible and programmable date back to the 1990s. Programmable Asynchronous Transfer Mode (ATM) Networks [23], Active Networks [24] and Routing Control Platform [25] can be considered as examples for programming the network components. In the following years, network virtualization methods were developed [26]. The ability to create more than one virtual network on a physical device contributed to resource sharing. Then, the most important achievement in this period is revealing to OpenFlow [22]. The origins of OpenFlow is found in Stanford University [8]. At the beginning, this protocol was designed for researchers to test experimental protocols in the networks. Then, Open Network Foundation took control of the project and has released new features and versions since that time. Since OpenFlow makes it possible to communicate network with low level devices in a standard procedure, the development of network operating systems are affected positively. Examples

of effectively developed and used supervisory mechanisms are NOX, POX, Floodlight, Ryu, OpenDayLight and ONOS [27].

The fundamental idea of the SDN concept is to centralize the distributed control structure of conventional networks [4]. SDN decouples the processes of gathering data and controlling traffic tasks to the two different layers to make the network more modular and manageable. In this way, the distributed and self-rolled management paradigm is abandoned and the control is assigned to a new responsible layer. Network devices leave the control and routing tasks to the control layer and are turned into the simple devices that only redirect flows according to the received commands via controller. In control layer, there are two main responsibilities which are assigned to the controller unit. One is to receive data from network devices and response them what action to take, and the other is to collect information about network devices and links and present network applications with an up-to-date network picture. At this point, the 3-layered SDN structure is defined; data layer, control layer and application layer. In the application layer, which works on the top the layered structure that takes defined metrics and information from the control layer and provides business applications and solutions to control traffic and other services. Similarly, the control layer, which acts as the brain of the network can consist of more than one controller unit for more efficient and task-based services. The control layer works as the operating system between the layers at both ends. With this virtualization, the application layer working with a simple network image can present the programs for the required actions. The control layer transfers and executes predefined/automated rules to all network components, namely the data layer. In Figure 2.3, the structure of the layers and the connection types between them are shown. In addition, modularly positioned networking elements, tasks and applications in each layer are exemplified.

The 3-layered SDN approach links layers with two gates which are southbound and northbound interfaces. The control layer and data layer communicate with each other using a southbound interface which is used to transfer network status changes and packet forwarding messages that are obtained from data layer elements via OpenFlow-enabled switches. Additionally, the routing and various action messages which are determined

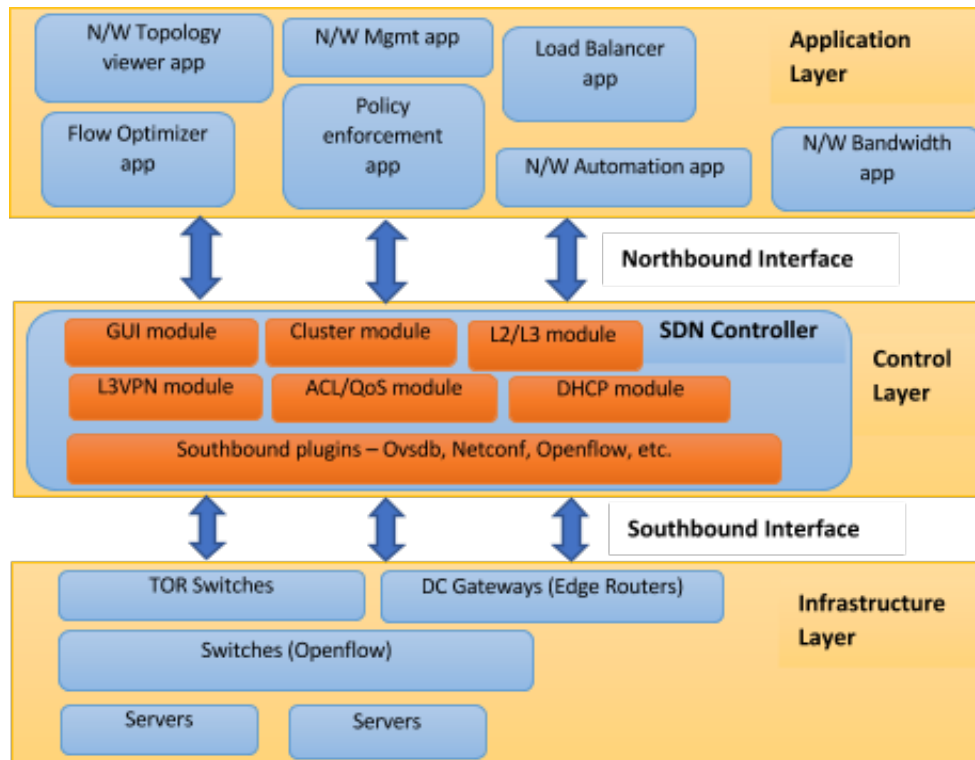


Figure 2.3 An overview of the 3-layered SDN architecture which is adopted from [4]

according to the network requirements are also transmitted to the data layer via this interface. The most preferred protocol in this interface is OpenFlow. On the other side, communication with the applications that are responsible for the dynamic management of the network is provided through southbound interface. High-level messages from the control layer and directives from applications use this gate which is not under the control of a single protocol.

In traditional networking, each routing device is responsible for forwarding datagram packets using a target-based forwarding approach. However, in SDN paradigm is not required the complicated switches that forward the packets according to the IP forwarding rules. In SDN, forwarding rules and procedures are defined around the flow-based approach. A flow consists of a series of IP packets going from the source device to the target device. There are various variables in the flow, such as IP address, VLAN tag, MAC address, that are used to implement this approach. Briefly, the routing and controlling approach made according to the data layer flow tables created by the controller with the dynamic decisions directed from the application

layer gives the network flexibility. In this way, flow-based routing is implemented instead of target-based routing.

The essence of the SDN is separated from the traditional networking by four fundamental features. These are separation of planes, flow-based routing instead of target-based addressing, separating the control into an external layer and redesigning the network to make it more programmable. These innovations bring effective benefits in terms of operational, financial and service quality:

- **Network business process cycle improvement:** Programmability and network automation features of SDN approach enable the agile improvement and proof of concepts for network engineers and administrators. The return rate of the investments are faster.
- **Open the way for innovation:** Modular structure makes it possible to try new methods, techniques, protocols and research for innovation enablers.
- **Fast response for customer requests:** SDN's modularity is capable of distributing network resources according to the users' demands. The dynamic allocation of the QoS needs on the SDN makes it possible to address customer requirements on the time.
- **Resource allocation and efficiency:** SDN infrastructure allows to determine intelligent or rule-based switching, routing, traffic engineering, network security solutions thus making it possible to use resources efficiently. These applications can be served as distributed and controlled via application programming interface (API).

2.5.1. Components of SDN

The OpenFlow and simplified layer based structure of the SDN make easy to manage and program whole networks in a centered way. The OpenFlow protocol is used to transfer some kind of information to control the network assets and status. When there is a link or a port

change is executed in data layer assets, the responsible or affected device sends the status message to the controller. Additionally, when a packet is received to the forwarding device in the data layer, it first looks for it in the flow table. and packet's routing information is searched through the flow tables. If no matching entries are found in the flow tables, the action can be defined as querying the controller for routing action, dropping the packet or no action. In the case a match is found for the received packet, the action is executed as a forwarding process. Simultaneously, the control layer transmits a snapshot of the network to the application layer. Business applications which are running in the application layer determine the operation structure of the system according to the network requirements. All these options depend on the directives of high-level programming. Following subsections, the roles and functionalities of the layers are explained in detail.

2.5.1.1. Data Layer SDN decouples getting data and routing packet processes. This phenomenon simplifies the responsibilities of the edge network devices such as routers and switches. Thus, data layer components that are converted to the more basic devices reduce costs. After the simplification, the controller communicates with each forwarding device on a standard protocol and assigns routing rules more homogeneously if it is required.

2.5.1.2. Control Layer The controller creates rules to be applied to the forwarding devices and provides the management of the network with the help of the OpenFlow protocol. In computer networking, each generated packet has some information in its header which are destination-source IP, port and MAC addresses, VLAN ID and priority, protocol and timestamp. The controller decides what to do with the flow when the datapath action type is requested by the data layer. Action rules are recorded to the flow tables in the network devices in order to make the routing operations faster and reduce the workload of the control layer. Thus, some flows can be directed or dropped by the network device without asking the controller. The controller has the authority to edit these flow tables according to the changing tasks over time and network requirements. On the other hand, centralizing management with

general network status information simplifies the development of more complex network functions, services and applications.

2.5.1.3. Application Layer SDN provides a centralized management which is assisted by the application layer. By sending the rules that are created in the applications to the network devices with the help of the SDN controller, the data layer devices constantly update their flow tables. High-level applications which implement control logic forward rules to control layer where instructions are converted into forwarding rules and installed on the underlying low-level hardware infrastructure. Using these layer possibilities, network administrators and traffic engineers have developed applications that serve many purposes.

2.6. Quality of Services

QoS approach aims to serve allocation of resources according to the network demands for each flow and user demand. Today's various network applications require distinguished QoS needs. Thus, the required service metrics are defined around 4 fundamental networking parameters which are bandwidth, jitter, delay and loss rate. With these measurement definitions, the QoS response the networking needs more elastically and dynamically. For example, a video conference call requires instant data transmission is more important with minimum latency. On the other hand, the delay factor can not be as important for data transfer, where data integrity is more important such as time series telemetry data. A well defined QoS allocation mechanism in networking can provide a sufficient service for both situations at the same time but the difficulties of the problem pulls the researchers to improve methods day to day [11].

In traditional networks, various QoS providing approaches have been developed to provide QoS requirements to the network users. With the over-provisioning method, the network infrastructure is created with high-capacity hardware and links. This has an impact on providing sufficient bandwidth and lower latency, but increases the setup and maintenance costs. On the other hand, routing elements are determined to provide different QoS

requirements in the differentiated service approach. However, the diversification of network devices complicate the system management.

The decoupled routing and forwarding approach in networking via SDN and OpenFlow has revealed various programmable structures for QoS. Queuing and bandwidth management are effective tools for the QoS allocations. With the help of SDN, flows can be queued according to their importance and rule-based bandwidth allocation can be made. Similarly, with resource allocation methods, packets can be classified in a central way and forwarded to fixed network resources. The allocation of the bandwidth resources are applied via predefined mathematical approaches as rule-based or learning based structure. Bandwidth management includes traffic manipulation, time-based prioritization and bandwidth restriction or relaxation techniques to optimize network flow traffic. One another method is priority queuing that sorts and processes the previously classified packets under a special policy. In this way, the packets that are prioritized with the definition of the need are served according to the routing and forwarding processes. Prioritization criteria can be defined around application, user and packet origin. Afterwards, it is tried to ensure that applications work as desired with bandwidth management techniques.

It is obvious the importance of OoS in the networking to manage and sustain a dynamic network environment. Decoupling data and control roles and easy programmability makes the SDN paradigm more advantageous than traditional networking. Scalability, separation of roles and flow-based routing provides elastic and dynamic allocation of the network resources for QoS which is defined on a SDN-based network. There are various studies in the literature to provide QoS on SDN [11, 28].

2.7. Mininet

Mininet is one of the networking emulator that works on Linux operating systems [29]. It provides a test bed that enables adding hundreds of networking nodes to virtualize a topology which includes virtual hosts, switches, controllers, and links on a single computer or a server. The virtualized network topology can be used to simulate both conventional and SDN-based

networking approaches to rapid prototyping.

As a networking emulator Mininet executes the networking emulation in real time. The networking emulators use real packets between network nodes to simulate actual traffic scenarios. This is the difference point between the networking simulators and emulators. Thus, while an networking emulator is running a test, the gathered status data is suitable for real-life analysis.

Mininet is a topology-aware emulator which can interact it with command line interface for running network-wide tests. While using Mininet, a controller can be implement from scratch or using an open source controller which have to be run with the Python API.

2.8. Ryu Controller

Ryu Controller is an open source SDN controller designed to increase network agility by facilitating control and management of network traffic [5]. As seen in Figure 2.4, The SDN Controller is the brain of the SDN environment, transmitting information with southbound APIs to switches and routers and communicates with applications and business logic via northbound APIs. Ryu Controller is supported by Japan Nippon Telegraph and Telephone and OpenStack which is a widely deployed open source cloud software. Ryu community maintains its source code on GitHub and is licensed under Apache 2.0. Thus, Ryu, written entirely in Python, is open to any organization and individual to use and supports OpenFlow, one of the most common SDN communication standards.

Ryu Controller provides software components with well-defined API that make it easy for network developers to create network management and control applications. Distributed component paradigm helps organizations customize deployments to meet their needs. Network administrators can quickly and easily replace existing components or implement their own components to ensure that the underlying network can meet the changing demands of their applications.

Network administrators can develop network and business applications that communicate with the Ryu controller on how to manage the switches and router to manipulate routing

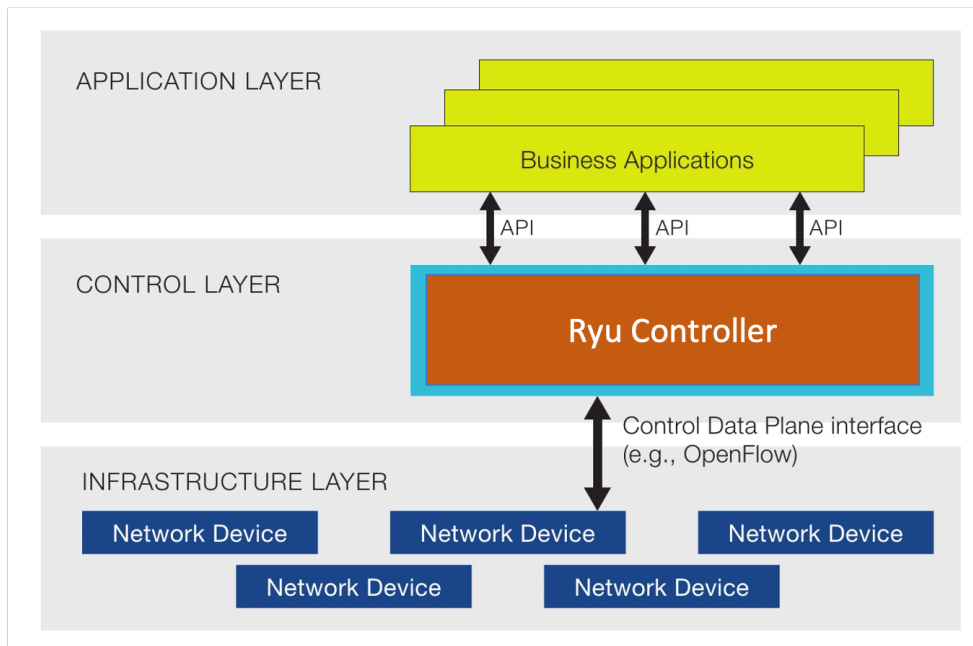


Figure 2.4 Overview of the SDN architecture with Ryu Controller [5]

and forwarding decisions in intelligent way. The Ryu controller can use OpenFlow or other protocols to interact with the data plane (switches and routers) to change how the network handles traffic flows. The Ryu controller has been tested and proven to work with a range of OpenFlow switches, including HP, IBM, and NEC products.

3. RELATED WORK

OpenFlow and SDN have significant accelerator effects on QoS and QoE requirements to design efficient routing decisions in networking. Decoupling the management of data and control processes, and programmable architecture at a single or distributed controllers are core factors for this paradigm [30]. The main QoS requirements can be vary by applications or services, but delay, throughput rate, jitter and bandwidth availability are most known and desired consistent service factors for end to end service experience. In research, the optimization of these service factors according to the application needs and network capabilities maintains its importance for many years [31]. These researches range from wide area networks (WAN) such as Internet level [32] to next generation technologies such as IoT networks, Cloud and 5G. [33–35]. In other side, the researches spread on the objective side and its methodological approaches. Some researches are focused to present a better solution for one QoS factor such as minimizing delay [36] or maximization of throughput [37–39], allocating available bandwidth [40]. On the other hand, some researches aim to improve on multiple factors of QoS such as renting service costs and balancing latency [41], minimizing delay and maximizing throughput [14, 42], delay-bandwidth constraint [43]. Additionally, researchers are able to follow deterministic, stochastic or hybrid methods according to the complexity of the problem [44–46]. In all these respects, we have grouped our literature section according to the number of QoS metrics targeted by the studies. In this way, preferred network environments and methods are given in details.

Later in this section, firstly Table 3.1 is given to summarize the some important points of reviewed works and then the details of them follow. At the end of this section, the critique of the recent literature and different points of this work are explained.

Ref. No	Year	Description	Objective	Technical Approach
[36]	2016	the brute force approach algorithm find paths and pairs with calculated end o end delays and then forwards lowest delayed path	minimizing delay	a brute-force search
[37]	2016	the algorithm considers a cost model by studying the admissions of unicast and multicast requests	maximizing throughput	approximation alg. for directed Steiner problem
[38]	2017	forecasts future link consumption using the ML model and generates forwarding rules	maximizing throughput	long short-term memory, recurrent neural network
[39]	2017	the algorithm searches paths and a randomized rounding mechanism to solve the multi-commodity h-splittable flow routing problem	maximizing throughput	depth first search
[40]	2018	ML model classifies traffic then the application class is used as inputs to the routing policy generator module to determine the distribution of traffic flow	bandwidth allocation	Nearest Centroid, Naïve Bayes, SVM
[41]	2018	the algorithm optimizes unstable network changes and minimize the latency use of dedicated short range communication	minimizing service costs and latency	improved genetic algorithm
[14]	2015	the approach classifies flows and forwards them according to the predefined QoS needs	minimizing delay and maximizing throughput	application classification and multipath routing
[42]	2018	proposed model distributes traffic via scheduling algorithm using network status	minimizing delay and maximizing throughput	a scheduler, probabilistic matching approach
[43]	2016	the algorithm makes dynamic routing to utilize network resources	bandwidth allocation, minimizing delay	custom flow-migration algorithm, Yen's algorithm
[47]	2018	the algorithm uses delay factor to create flow rules from gathered statistics	minimizing delay	shortest path algorithm
[48]	2018	the algorithm calculates delay statistics using proposed approach and forwards the related flow to the lowest delayed path	minimizing delay	Timestamp Recording method, Dijkstra
[17]	2017	proposed approach makes aggregate operation to reduce total flow size and then control switches congestion problems	minimizing delay	flow aggregation and heuristic algorithm
[15]	2019	the algorithm first distinguishes flows as mice and elephant. Then the seperated flows are forwarded as predefined paths or routing rules	bandwidth allocation, load balancing	flow classifier approach
[16]	2018	the approach finds available paths from source to destination and forwards flows using proposed method	bandwidth allocation, minimizing delay	Dijkstra, rule-based flow forwarder
[13]	2012	the framework categorize incoming flows multimedia and others then forwards them based on their QoS constraints	minimizing delay packet loss, bandwidth allocation	Lagrangian Relaxation Based Aggregated Cost
[44]	2017	the framework adaptively adjusts weights of QoS metrics in a cost function to find the best fit path according to QoS requirements	optimizing delay, loss rate and bandwidth	simulated annealing
[49]	2020	the approach utilizes ML algorithms to select the least congested path from a list of possible paths	bandwidth allocation	K-means clustering, cosine similarity
[50]	2016	proposed algorithm forwards flows according the reward function maximization to meet QoS requirements	minimizing latency, bandwidth allocation	Q-reinforcement learning
[51]	2016	a multi-layer hierarchical SDN control framework, their method calculates routing paths that maximize the network QoS performance for a given traffic type	optimizing delay, loss rate and throughput	Q-reinforcement learning
[52]	2017	the approach defines the route using RL according to the best criteria in terms of weights periodically rewarded by each node on the current path	optimizing delay, loss rate and bandwidth	Q-reinforcement learning
[53]	2017	the algorithms attempt to minimize end to end delay on its route decisions that achieved by using current traffic conditions on RL agent	minimizing delay	deep reinforcement learning
[54]	2016	the proposed algorithm finds the optimal integral routing solution for overlay network of data centers	minimizing delay	random neural nets with reinforcement learning
[55]	2021	the framework utilizes RL to select optimal for all flows with the QoS metrics	optimizing delay, loss rate and throughput	Q-reinforcement learning
[56]	2021	the approach selects best pre-build forwarding protocol to route flows according to the QoS needs	optimizing throughput, loss and rejection rate	Q-reinforcement learning
[57]	2020	the proposed method analysis time series network data with heuristic algorithm and find ideal routes	load balancing	linear programming, fuzzy logic
[58]	2019	the proposed method get information from past traffic demands and optimize the routing policy to meet QoS needs	minimizing delay, improving net security	deep reinforcement learning

Table 3.1 QoS factor optimization centered researches with details

3.1. Single-factor centered QoS approaches

A rule-based implementation of routing decision is proposed in [47] which uses delay factor to create flow rules on the controller. Their approach collects link statistics periodically from data layer elements to the controller and a delay weighted Dijkstra algorithm searches the least delay occurred path. Proposed framework is compared with unweighted Dijkstra and L2 Switching. Another early effort for delay-optimized work in an IoT is proposed in [36]. The method aims to find the route which has minimum delay. The algorithm first finds suitable paths and assigns nearly up-to-date link delays via OpenFlow and selects with the minimum path delay. Chin et al. compared 3 different delay measurement approaches to make efficient traffic engineering while they were forwarding flows to minimum delay paths [48]. Making a routing decision based on only one QoS factor can prevent you from seeing the big picture of the network. [42] offer a dynamic traffic engine to schedule delay sensitive optimized flows. Their algorithm uses an auxiliary graph that is fed information retrieved from data plane to create multi-path flows. One of the most complex method is demonstrated in [17] that tries to handle the problem of flow table overflow in switches. To the that, a flow aggregation technique is provided to minimize number of flows on the network when satisfying end to end latency. Flow aggregation is not in conflict with flow splitting, but sending aggregated flows together can issue for objectives such as load balancing. At this point, Shi et al. proposed an load balance-aware controller to manage network traffic with an efficient bandwidth usage. Their method offers for different forwarding rules for mice and elephant flow requests [15]. However, the proposed fixed route forwarding policy for elephant flows can cause some local congestion because of the stochasticity of the traffic. Additionally, proposed rule-based load balancing strategy for switches can suffer on a series of elephant flow requests. One of the most preferred approaches in effective traffic management with SDN is based on load balancing control [59]. Nkosi et al. offer load balancing approach for data plane to optimize link utilization according to the link loads [16]. Basically, the framework finds paths with Dijkstra algorithm and forwards flow to the lowest loaded one and shared fairly among the alternative routes.

3.2. Multi-factor centered QoS approaches

Egilmez et al. [13] proposed a highly potent framework, named as OpenQoS which handles traffic as multimedia and data flows. Multimedia flows are forwarded by QoS guaranteed routing algorithm and data flows follow the conventional shortest path algorithm. Split-centered protocols can be inadequate to provide required QoS factors to the left behind flow classes. Another same flow classification approach is proposed in [14] as normal data flow and multimedia flow for the incoming flow request to the controller. HiQoS has designed in two components, differentiated and multipath routing. Differentiated component dissociates incoming flow requests according to their service types and provide predefined bandwidth guarantees to them. Then, multipath component calculates optimal multiple paths for each flow that meets the QoS constraints of the service types. Their approach reduces delay and increase throughput in the simulations. However, a bias flow distinguish procedure can miss mice and important flows' QoS requests. In [44], authors offer a simulated annealing based method to find best path according to the three QoS factors, delay, bandwidth and loss rate. Their approach defines a cost function with these factors with weights. The weight of each factors are adjusted by simulated annealing method which tries to meet the QoS requirements for each flow request. One of the main advantage of the simulated annealing is to beating the local minima traps but in a large scale network topologies, the high computation time is required to guarantee to achieve a qualified outcomes. An SDN and ML based attempt explores to route flows via congestion and traffic pattern history of the network [49]. They proposed two methods, k-means and Vector Space model. In framework, the training unit learns updated network-link bandwidth patterns from network statistics simultaneously. The given algorithms put paths to existing clusters, or the clusters are rearranged to form a new set of clusters and the best paths for each states are defined by the network admins during the training. In deployment unit, when a flow request arrives to the controller, the network's bandwidth states are collected and provided as input to the ML model. The best path is selected from the cluster which is suited for the input. Their approaches surpass Dijkstra but one point is that clustering algorithms such as k-means are not suitable for large datasets, in this case large-scale nets. [50] tries adaptive

routing with Q-learning algorithm. They offer a reward function that is a weighted sum of bandwidth availability, number of flows in the next hop and latency rate. Rewards are used as signals to adjust referral link priorities to increase or decrease the probability that a particular next hop will be selected for traffic. An agent learns to adjust path selection policies based on experience and rewards and tries to maximize some cumulative returns through continuous modification of action selection policies. Using the routing connection probabilistically based on the Q values in each switch to choose the most appropriate end-to-end route in network traffic for each available path can bring computational cost in long path determinations. Lin et al. [51] offer an QoS-aware adaptive routing (QAR) in a multi-layer hierarchical SDN control plane. Their SARSA-based RL approach manages the weights of the QoS factors such as loss, throughput and delay in the reward function. In this study, they recommend that the controller need to iterate the SARSA algorithm. This causes delayed flow initiation and its suitability for real-time traffic is questionable. Sandra et al. [52] propose an RL-based controller that simultaneously considers delay, bandwidth, and loss to optimize QoS. When a flow is requested from network, the trained model offers weights for each QoS factors on every possible links and then lowest cost path is selected. Then, network feedbacks the controller based on the current path as a reward or penalty. Such a link based approach is likely to cause local forwarding biases in networks that have unequal bandwidth distributions. One another study targets the minimizing only delay metric in flow forwarding when offer a Deep RL approach [53]. In [54], Random Neural Networks and RL methods are used to build a SDN-based cognitive routing algorithm. Their approach aims to reduce delay factor and efficient routing management for cloud data centers. A complex deep reinforcement learning based approach offers predicted next hops when tries to guarantee QoS needs. Another recent work in [55] designed a RL-based routing algorithm. Their approach uses RL to find best route for all flows with bandwidth, loss, and delay. Then, to define best path, it searches most-rewarding path for every pair of nodes in the topology. One of the latest cognitive SDN controller attempt offers to route flow requests according to the selected routing protocol from protocol pool based on the instant reward of the their Q-learning method [56]. Their reinforcement learning framework try to select best route for each flow according to the throughput, delay and rejection rate. In [57], authors propose a

load balancing approach to meet multiple QoS requirements in IoT servers. They offer an integer linear programming and tries to solve the problem with fuzzy logic approach. An intelligent and secure routing protocol is proposed in [58] as a deep RL based for IoT-SDN infrastructure. The algorithm extracts knowledge from traffic history using the trained model and interacting with the underlying network environment, then optimize routing policies according to the QoS requirements.

When we analyse the previous works, we see that many different optimization approaches to meet QoS needs have been proposed. These approaches are shaped around deterministic and stochastic solutions. The network traffic environment is very variable. Thus, it is not possible to provide an effective QoS in all situations with rule based approaches. In this context, ML techniques can make effective routing results in non-linear network traffic conditions. For this reason, reinforcement learning method was preferred in our study. Another situation seen in the literature is that when the QoS needs of the flows are taken into account, the effective use of network resources is left in the background. However, routing each flow around the network capacity according to the load conditions of the switches can be a more democratized solution. Therefore, in our study, we aim to optimize the QoS requirements for each flow with a reward approach aiming to keep the network load balance optimal base with the q-reinforcement learning proposal that needs low computational power. To the best of our knowledge, no previous study has been carried out in this context.

4. PROPOSED METHOD

This section describes the proposed techniques to design a routing method in the study. In order to meet the QoS requirements for flows and also to perform load balancing on the switches, periodical monitoring of the amount of data passing through the ports between the switches is required. Unlike the traditional networking approach, the route searching task is transferred to the control layer in the SDN. SDN allows to develop tools that can manage all data layer elements. Thus, data layer networking devices do not determine the routing for a flow request to be established for the first time. In this study, a route discover module is developed in the control layer. This module finds all routes between two hosts via a searching algorithm. Searching for available routes to meet the flow requests and tracking the traffic statistics of the data layer elements are detailed in Section 4.1.

In this work, three different network metrics are measured in order to provide the most optimal QoS to any flow requests. These factors are determined as the available bandwidth of the route, the delay time of the route and the total number of hops of the route for the selected routes. For each flow request which has not been routed before, route discoverer module finds predefined number of suitable routes between source and destination. Subsequently, the above-mentioned route state metrics are calculated for each route. These instant QoS metric tracking mechanism is discussed in Section 4.2.

Section 4.3. which is the core part of the study, decides which route to use for any flow request. We prefer to propose two different methods in order to make performance comparisons. The first is the equally weighted QoS approach and the other one is the RL based weight determination QoS method. Firstly, the total number of hops, available bandwidth and total delay time for each candidate route is normalized between 0 and 1 in order to prevent any dominance of metrics' values. In the equally weighted QoS factor approach, which is the first of the approaches, each performance metric is multiplied by a weight of 0.33 to contribute equally to the QoS score. In the other approach, QoS factor weights are defined by taking into account the load balancing with the Q-learning algorithm.

Prior to this approach, the system is trained with RL approach with an extended network traffic scenario and a Q-table is created. Then, the QoS score is calculated by obtaining the appropriate weight values from the pre-trained Q-table for the route that is most likely to balance the system according to the loading status of the switches. In both approaches, after the calculations, a QoS score is obtained for all suitable routes and the route is selected as the routing path that has the maximum QoS score. A detailed description of the proposed approaches is discussed in Section 4.3.

After determining the appropriate route, the flow starter module performs the placing of the selected route in the switches' flow tables. It issues the flow rules as messages via the OpenFlow protocol from the control layer so that route details can be recorded to the flow table of each switches on the selected route path. The method created for the routing module is discussed in Section 4.4.

4.1. Network Traffic and Route Awareness Module

In order to design a traceable and manageable network system, the data layer elements on the topology have to be discovered in the first place. With the awareness of each network element, packets can be transmitted via an intelligent routing techniques on determined routes. In addition, traffic engineering applications can be developed by monitoring the current load intensities of network elements and their links. In this section, the methods presented to realize explained features.

4.1.1. Route Discoverer

Graph theory is widely researched in computer science and applied mathematics. A data structure consisting of vertices and links connecting them (optionally directed/weighted) can effectively represent many problem areas [60]. The topological structure of computer networks coincides with graph theory. The problem of routing the flow requests, which is also a problem of the Internet, is solved by search algorithms at various levels. Two of

the most well-known search algorithms which qualify as unweighted Dijkstra to find the available paths are Depth First Search (DFS) and Breath First Search (BFS) [61]. The BFS algorithm is preferred for the shortest path first return guarantee. Another side, DFS has a first depth strategy that searches for possible vertices on each branch before tracing back. The pseudo-code of the defined searching algorithm is given Algorithm 1.

Algorithm 1 BFS Algorithm (taken from [61])

Require: G is whole topology
Require: s is source vertex
Require: t is destination vertex
Ensure: $stack = (s, (s))$ ▷ Let S be a stack. Insert s in stack
while $stack$ is not empty **do**
 $v = stack.top()$ ▷ pop a vertex from stack to visit next
 $s.pop()$
 for all neighbour w of v in Graph G **do** ▷ push all the neighbors of v in stack that are not visited
 if w is not equal t **then**
 yield $s.insert(w)$ ▷ insert w to the s
 else
 $s.insert(w, (s), w)$ ▷ store all routes in s

When a flow request message from any data layer element reaches to this module, optimal routes between the source and destination switches are found and listed using BFS. Then, the routes are forwarded to further modules that are defined in Section 4.2.

4.1.2. Network Statistics Tracker

The controller basically monitors and manages the network. Problems such as latency and data loss are often associated with congested connections and busy switches. For a sustainable and stable network management, a control unit design is required that continuously measures the system elements and plans the network actions according to the status of these elements. The provided sub-module collects metadata (almost real time) about packets passing through each switch's ports within a specific period of the time in order to measure the load status of data layer elements. With this information, the total load graph of

the switches and the amount of instantaneous data passing through the links are measured. Therefore, applications such as available bandwidth and load balancing of interconnects with known capacities can be developed. As shown in Table 4.1, the details of the information obtained through each port of each switch on the network are given.

Table 4.1 Data types that collected periodically from each port on switches

Abbreviation	Description
switch-id	Switch Unique Number
port-id	Port Number
rx-pkts	Received Packet Count
rx-bytes	Received Bytes Count
rx-error	Received Error Count
tx-pkts	Transmitted Packet Count
tx-bytes	Transmitted Bytes Count
tx-error	Transmitted Error Count

The instantaneous packet and data statistics obtained through this module are used as input from the next modules to determine the appropriate bandwidth on the links between switches and to determine the total load on the switches.

4.2. QoS and Load Awareness Module

There are several network performance factors that can be used for traffic engineering with QoS aware on a network. The priorities of traffic requests created by users and hosts on the network are usually centered around QoS criteria such as latency and bandwidth. Therefore, available bandwidth tracker and latency time tracker modules have been developed for a QoS-aware routing. In addition, we offer as a new QoS factor, the number of hops, which is the basis of the best effort approach, which is also included in the QoS metrics. Additionally, in our RL-based routing approach determines routing decision based on load status of switches to provide robust network management while meeting QoS needs of flow request.

For that reason, a switch load tracker module is given under this section in order to make a load balance-aware routing according to the network load status using proposed RL-based approach.

4.2.1. Available Bandwidth Tracker

Detecting the availability and congestion of the links in the network allows for more effective system monitoring and traffic engineering. Balancing the data load from higher to the lower side on network elements is a routing strategy for load balancing [16]. With the help of the network statistics collection module, the load status on the switches and the appropriate bandwidth of links can be measured.

The appropriate bandwidth monitoring module basically consists of two parts. The first part periodically measures the available bandwidth of all connections on the network. Each link between switches has symmetrical bandwidth capacity. The amount of data passing through all ports are obtained from the network statistics module. The amount of bandwidth available for the link is found by subtracting the amount used at time t from the default bandwidth capacity of the link. This process is repeated for all links periodically and the available bandwidth of the entire network is mapped which is used to feed the RL approach for the environment status as load status of the switches. The design of the available bandwidth tracker is presented in Figure 4.1.

The second part is responsible for calculating the available bandwidth for each of the routes transferred from the route discovery module. Each transferred route information to this part includes switches and its port-link details. With this information, the available bandwidth of the links is measured using bandwidth tracking module. Since the smallest available bandwidth of all links of the route causes a bottleneck for the route traffic, it is determined as the available bandwidth of the route. Then, this process calculates the available bandwidth for all incoming routes'. The working design of this part is given in the Figure 4.2.

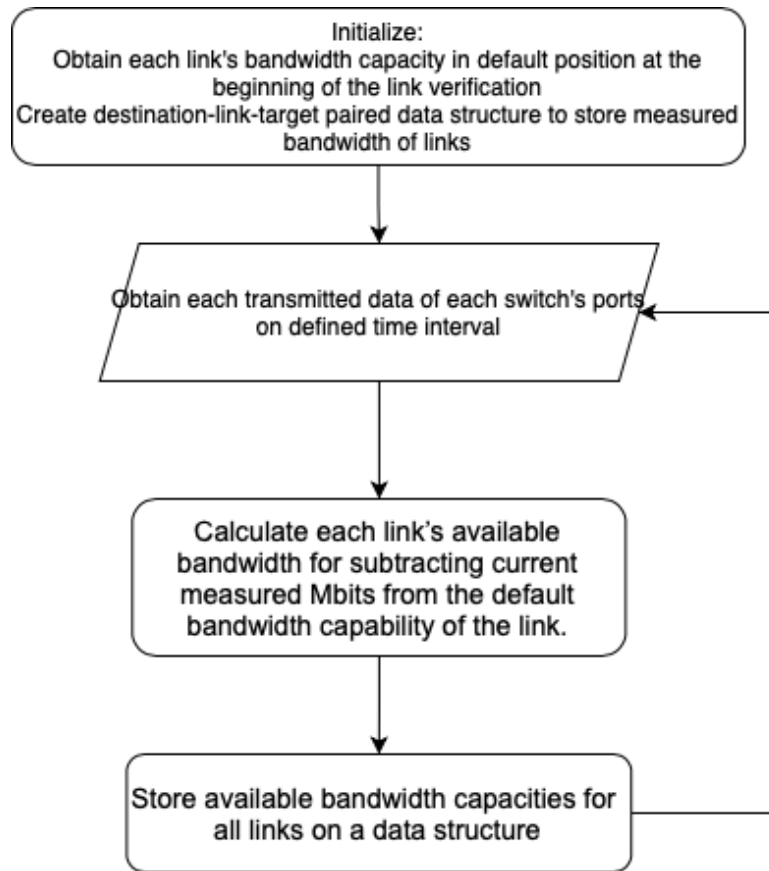


Figure 4.1 Flow diagram of periodical bandwidth tracking process

4.2.2. Delay Time Tracker

Periodically obtaining the delay time of the switches and links on the network is important for a stable and manageable QoS aware application design. It is critical for the delay-care applications where instant communication is important (teleconferencing, online gaming, fintech applications, IoT etc.). Therefore, a transparent latency measurement of the network environment is required for better routing decisions can be made and to meet the QoS and QoE demands. The delay time tracker module basically consists of two sub-parts. One of them is periodically measures delay time of the links. It runs as a background endless job and records the propagation delay time on the links between two switches and additionally the transmission and processing delays of these switches. The second part of this module performs the task of calculating the current total delay time for the routes which transferred from the route discovery module. The proposed method for calculating the packet delay time

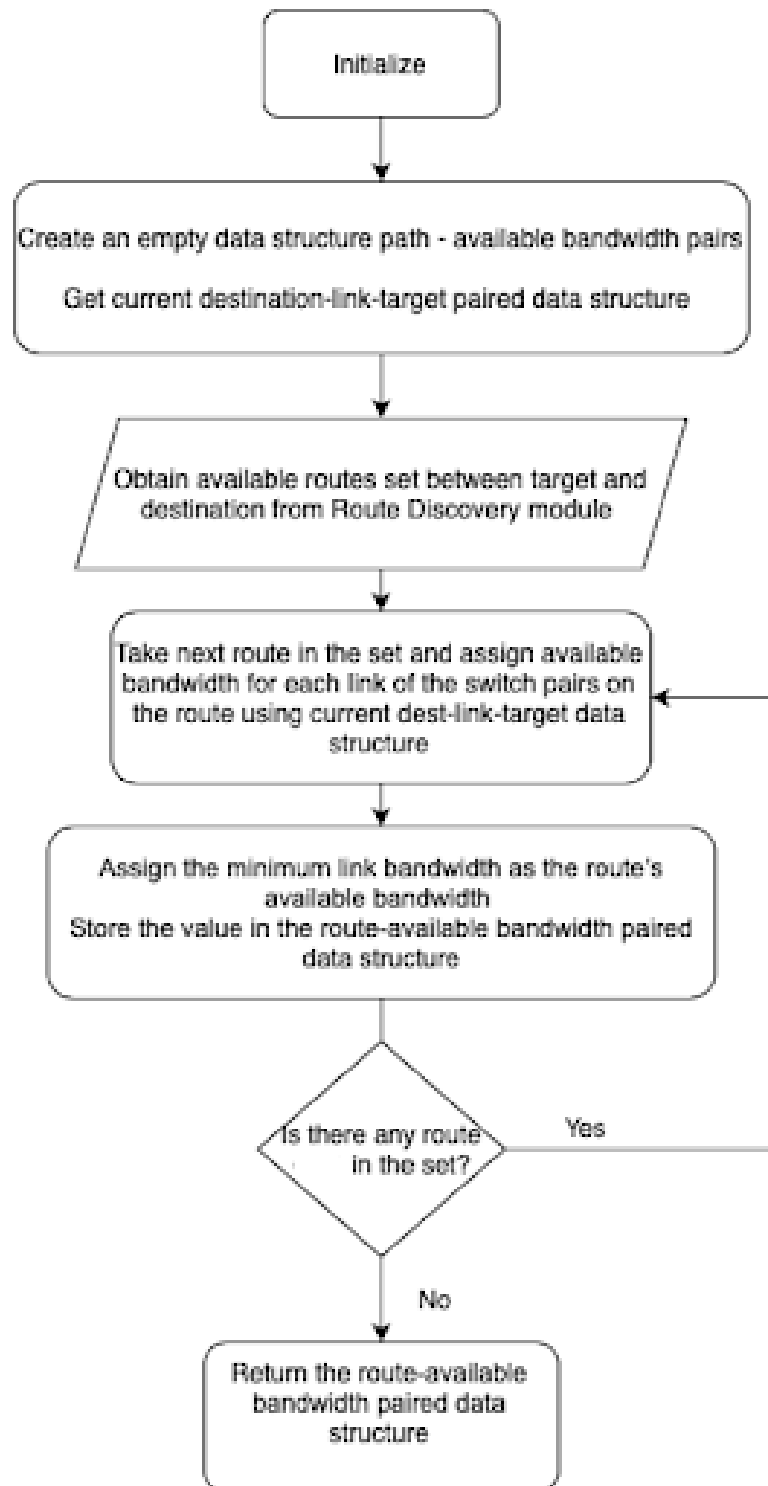


Figure 4.2 Flow diagram of available bandwidth determination process of a route

of each links with a certain time period on the network elements is shown in Figure 4.3. In this way, the delay time between each switch-link-switch tuples on the network is kept up to

date.

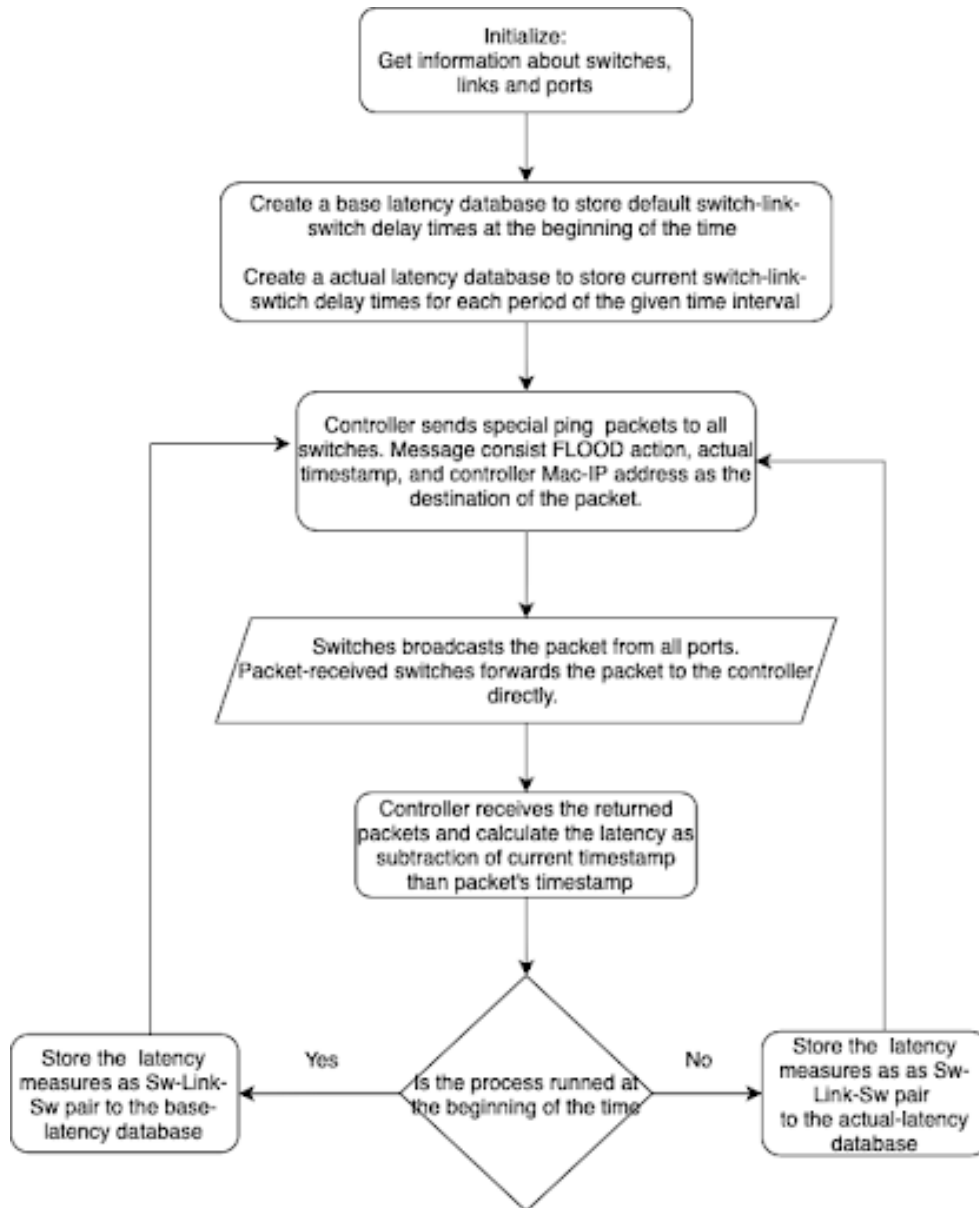


Figure 4.3 Flow diagram of periodical delay time tracking process

In Figure 4.4, it is demonstrated how the total delay time is calculated periodically in a topology that contains 3 switches. The method follows the procedure suggested in [62] and is implemented via OpenFlow with Link Layer Discovery Protocol (LLDP) packets issued by the controller [63].

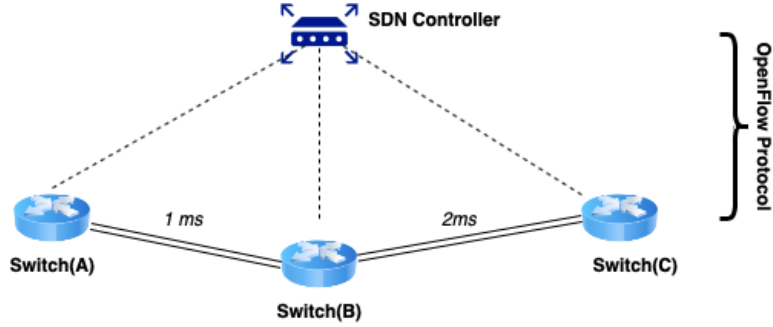


Figure 4.4 An illustration of the delay measurement procedure in a mini topology

According to Figure 4.4, the controller c_0 constructs an LLDP message with current time and sends it to the switch that is named sw_B in this case. It sends this packet all ports except from received port. sw_B broadcasts the packet to its neighbour switches which are named sw_A and sw_C . They forward the received message immediately to c_0 via controller-assigned port. c_0 calculates actual timestamps for each returned messages from sw_A as $D_{lldp_{sw_{BA}}}$ and sw_C as $D_{lldp_{sw_{BC}}}$. The time taken to go c_0 to sw_B is accepted as half of the OpenFlow echo request as D_{echoc_0B} . Similarly, this procedure is estimated for taken time when the message is forwarded from sw_A to the c_0 as $D_{echosw_{A0}}$. To be more descriptive, instantaneous link delay time between sw_B and sw_A is calculated as $D_{sw_{BA}} \simeq D_{lldp_{sw_{BA}}} - D_{echoc_0B} - D_{echosw_{A0}}$.

The second part of this module is responsible for calculating the total delay time for each received routes from the route discoverer module. Each received route has the path and link information between the switches. This part obtains delay times from previous part of the module for each link on the route. This process calculates total delay time for the route and the algorithmic flow is given in Figure 4.5.

4.2.3. Route Hop Tracker

RIP is one of the oldest distance vector routing protocols that uses hops as a routing decision [64]. The protocol prevents routing loops by imposing a limit on the number of hops allowed in a path from source to destination. At the Internet and local network levels, the hop number is used in traffic engineering studies and routing is performed with the least effort. Forwarding a flow request from the route that has the least number of switches has both

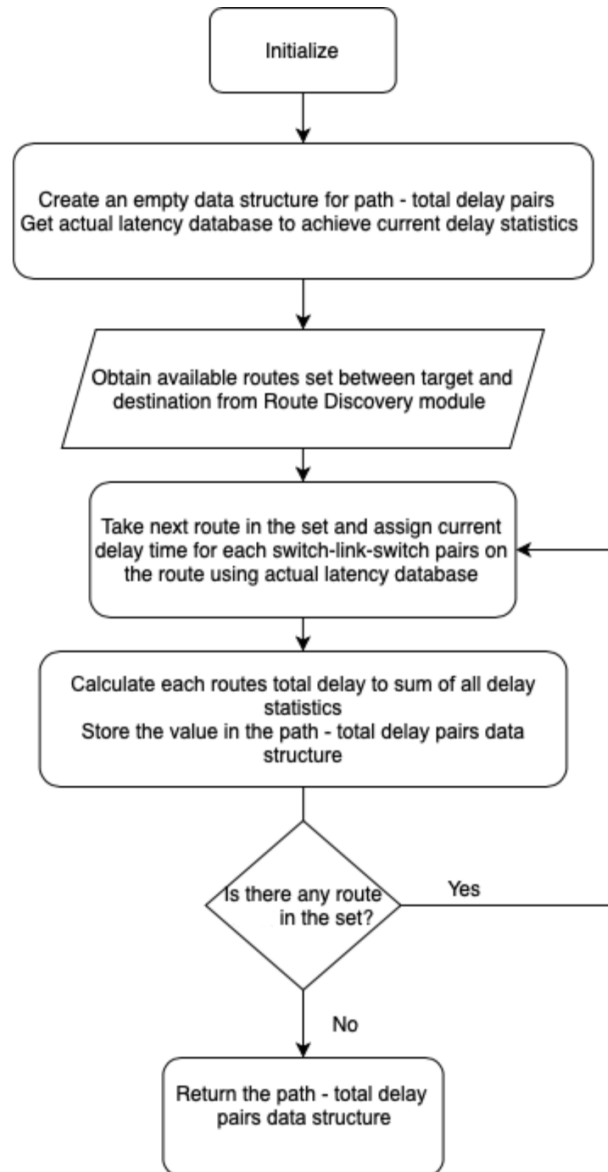


Figure 4.5 Flow diagram of total delay time determination process of a route

advantages and disadvantages. In normal conditions, a flow request that passes over fewer switches is exposed to less delays during switch transitions and is likely to be transmitted faster and with less packet loss. However, the reality may be negative for a situation where traffic has heavy density within the same network area. Since the protocol will transmit the new request on the shortest route regardless of the network load density, the congestion on the switches or links cause packet losses and delays. The decision of routing according to

the number of hops at the center of best effort routing has advantages and disadvantages. The hop number on the route is proposed as an QoS metric in our study to take advantage of the power of best effort paradigm. For this reason, a route hop count tracker module has been developed. This module calculates the total number of switches in each of the routes which are issued by the route discoverer module. The flow diagram of the module is shown in Figure 4.6.

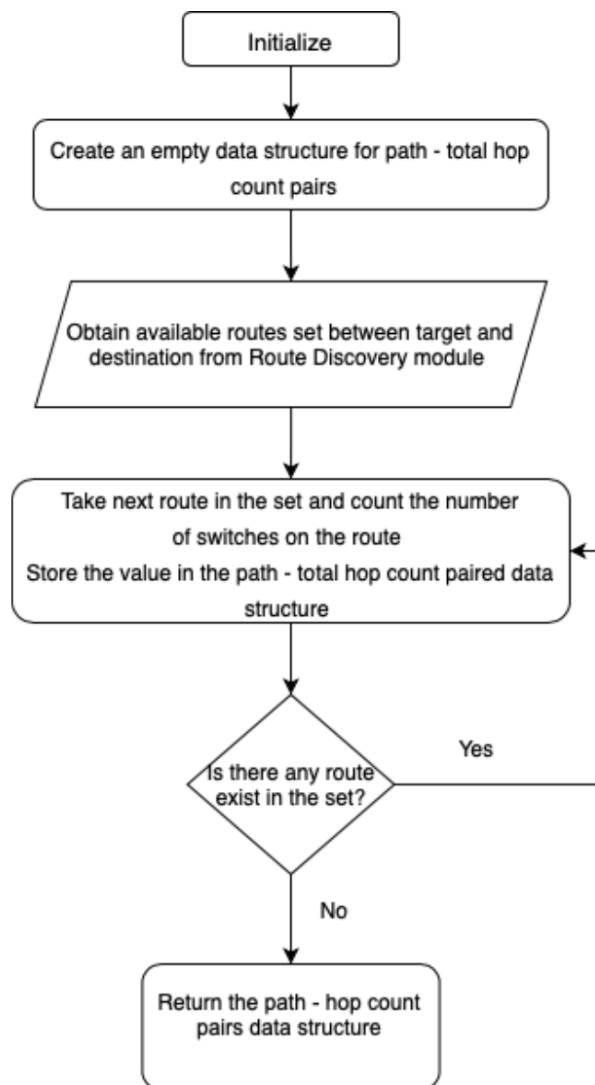


Figure 4.6 Flow diagram of total hop count determination process of a route

4.2.4. Switch Load Tracker

Considering only the QoS demand of the flow while taking the routing decision may cause uncontrolled load imbalances in the data layer elements, and it results in congestion problems. Therefore, delay and data loss problems are seen in the transferred data. A switch load tracker has been developed to establish a routing mechanism with the help of the overall load image of the network. It obtains periodically port I/O values from network statistics tracker and calculates data density percentages of the determined switches in the network, then this information will be provided as an input to the RL agent as the environment state in the next section. The load percentage of each of the switches is calculated as shown in equation 1.

$$L_{sw} = \frac{\sum_{i=1}^n (p_{rx-bytes}(i))}{\sum_{i=1}^n (p_{link-cap}(i))} \quad (1)$$

L_{sw} represents the load percentage of a switch at time t . The amount of data passing from each port $p_{rx-bytes}$ to the switch at time t is summed. The sum of the default data capacity of link in bytes that are connected to the each port $p_{link-cap}$ is obtained. The summations are continued around number of n ports for each switch. Then, the fraction of them gives load percentage of the switch.

4.3. Route Selector Module

The variety of applications that generate traffic on the network requires customized routing due to the QoS needs of the applications. In previous studies, it is seen that a routing decision is made according to the protocol of the application or the type of request (mice or elephant flow) [15, 52]. Prioritized applications or flow requests cause a large percentage of usage, while less prioritized flow requests stay in the background, causing various user dissatisfaction and QoE issues. Due to these shortcomings, a weighted QoS scoring has been proposed. A more load-balanced and democratized network management approach is offered with a routing approach based on the QoS score of the routes.

In the QoS and load awareness module, the available bandwidth, total delay time and total hop count metrics are calculated for each candidate route with the statistics obtained from the data layer and the route information obtained with the route discoverer modules. Using these metrics' weighted average sum, a QoS score is calculated to choose the most appropriate route. Subsequently, the route with the highest score is selected for forwarding decision. In the literature, [14, 55, 57] has similar weighted QoS approach with various methods. The same QoS scoring method is used in the 2 routing approaches under this module.

The number of hops, available bandwidth and delay time which have different scales and units contribute parametrically to the routing decision score. When a new flow is requested to the controller by data layer, route discoverer module finds available bunch of routes and forwards to the each QoS awareness modules. Each of these modules calculate their respective values and then transfers to the QoS score calculator module. In order to evaluate these variables in the same range and calculate a score, these are normalized according to their own spaces with a value in between 0 and 1 using the Min-Max technique as described in [65]. In this context, the normalized ratios of total delay time d_{route} , available bandwidth bw_{route} and hop count h_{route} are defined as \hat{d}_{route} , \hat{bw}_{route} and \hat{h}_{route} respectively.

$$\hat{d}_{route} = \frac{d_{route} - \min(d_{route})}{\max(d_{route}) - \min(d_{route})} \quad (2)$$

$$\hat{bw}_{route} = \frac{bw_{route} - \min(bw_{route})}{\max(bw_{route}) - \min(bw_{route})} \quad (3)$$

$$\hat{h}_{route} = \frac{h_{route} - \min(h_{route})}{\max(h_{route}) - \min(h_{route})} \quad (4)$$

QoS Score equation (5) is created as maximization formula with weighted summation of the normalized QoS factors. Equations (2), (3) and (4) are represented by an objective function that seeks for the maximum QoS score. Therefore, the best case for each variable occurs when it is close to 1.

$$QoSScore = W_d \times (1 - \hat{d}_{route}) + W_h \times (1 - \hat{h}_{route}) + W_{bw} \times \hat{bw}_{route} \quad (5)$$

(2) performs its lowest ratio when there is least delay in the selected route. Therefore, it is represented as subtracting it from 1 to reflect this in (5). Conversely, if the delay time of the route is large, the number converges to 1. In this case, its effect on the (5) is less. Similarly, the ideal scenario for (4) is when the number of hops are measured on the route is the least. Since it will converge to 0 in case of at least 1 switch, it has been subtracted from 1 to make a greater effect. In (3), the ratio gets closer to 1 as the available bandwidth expands. Therefore, it is directly represented in (5). In the opposite case, the ratio converges to 0 that results a lower contribution to the OoS score. Score calculation with weighted multipliers is common in the recent literature [55–57] as we defined in our QoS Score equation in (5). Similarly, they use weighted sum or difference method in the objective functions of their SDN-based approaches.

As shown in (5), each QoS factor ratio is also multiplied by a weight value. The value determination of the W_d , W_{bw} and W_h weights constitutes the main difference of the 2 routing approaches proposed in this study. While the Equally QoS-aware Route Selector determines the QoS factor weights as equally, Q-learning based Adaptive QoS and Load-aware Route Selector, that also offers a load-aware system by assigning specific weight values via Q-table.

In Figure 4.7, an overview is given for two routing approaches. The layer-based structure is preferred as known in SDN paradigm [4]. On the left side of the figure, equally-weighted QoS-aware approach is shown. We offer this approach to compare our main method with another SDN-based routing methods. The proposed Q-learning based approach is on the right side tries to predict adaptive weights to meet QoS needs and optimal switch load distributions. Details of these approaches are given in the following sections.

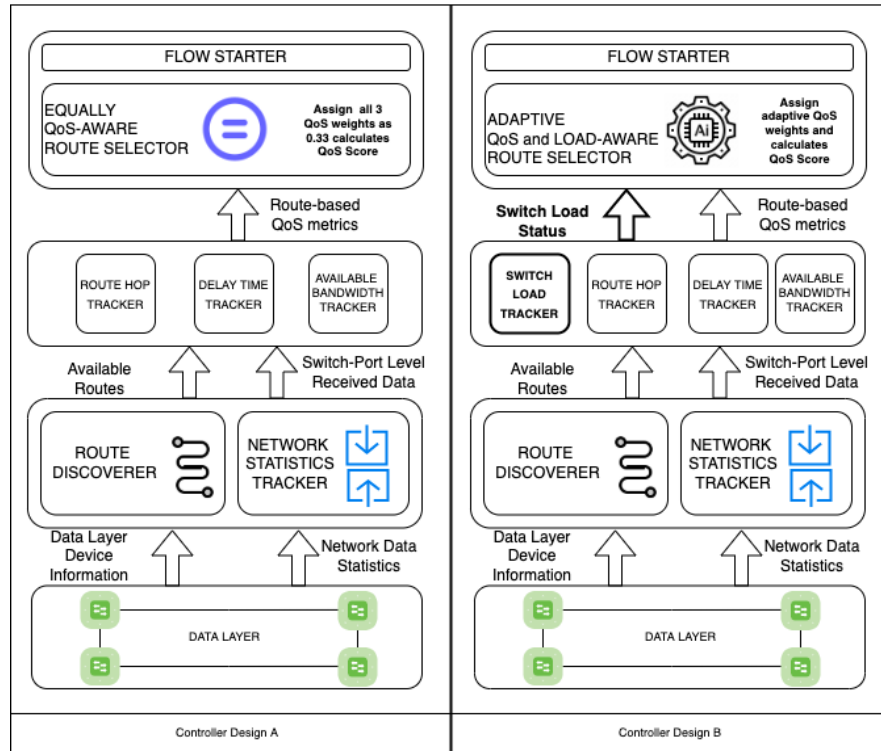


Figure 4.7 Comparative overview of the proposed RL-based controller (Design A) and rule-based controller (Design B) frameworks

4.3.1. Equally QoS-aware Route Selector

In this approach, it is designed to calculate the effect of each QoS factor in the objective function as equal. The purpose of this approach is based on the thesis that a reliable and robust network management can be created by representing the total latency, available bandwidth and total hop number factors with an equally weighting manner. The importance of each factor is considered equal in each flow request, then the highest QoS scored route is assigned as the appropriate route to the forwarding module. The predefined weight values are shown in Table 4.2.

After each new flow request is issued to the controller, the QoS factor ratios for the appropriate routes between the source and the destination are determined with the help of the QoS and Load Awareness Module. Then, in this section, QoS scores are calculated with QoS ratios and the assigned weight values for each route using eq. (5). At the end, the route that has the highest QoS Score is selected for forwarding the flow.

Table 4.2 Equal predefined QoS factor weight values

QoS Factor Weight	Assigned Weight Value
W_h	0.33
W_{bw}	0.33
W_d	0.33

4.3.2. Q-learning based Adaptive QoS and Load-aware Route Selector

Flow requests on the network have different QoS demands for their services, consequently it requires a customized QoS. On the other hand, a QoS prioritization without considering the load status of the switches on the route can cause congestion, delay and data loss. The main objective of our approach is to provide the most ideal QoS for each flow, while making an intelligent routing by considering the load balance of the forwarding devices at the same time.

4.3.2.1. Overview In order to establish an adaptive routing mechanism with Q-learning according to the load status of the system and the QoS needs of the flows, the framework have to be defined in the context of computer networks. In the Q-learning approach, the current status of the network is observed by the agent. According to the observed network load situation, the agent tries to choose the optimal action corresponding to the current situation from a random action or Q-table and applies it to the environment. In this context, the action is to determine the weights for the QoS factors. Then, system calculates the QoS scores for the appropriate routes with the provided weights obtained from the action. The route with the highest QoS score is found and the related route is established in the data layer switches' flow tables. This action starts the demanded flow on the data layer and the load status of the data layer elements is changed. For the new state, the environment gives to the agent a reward or a penalty score. This feedback updates the Q-table with the Q-function to account for future rewards. This cycle continues by feeding itself within the system. In Figure 4.8, our proposed routing approach is roughly illustrated in RL cycle.

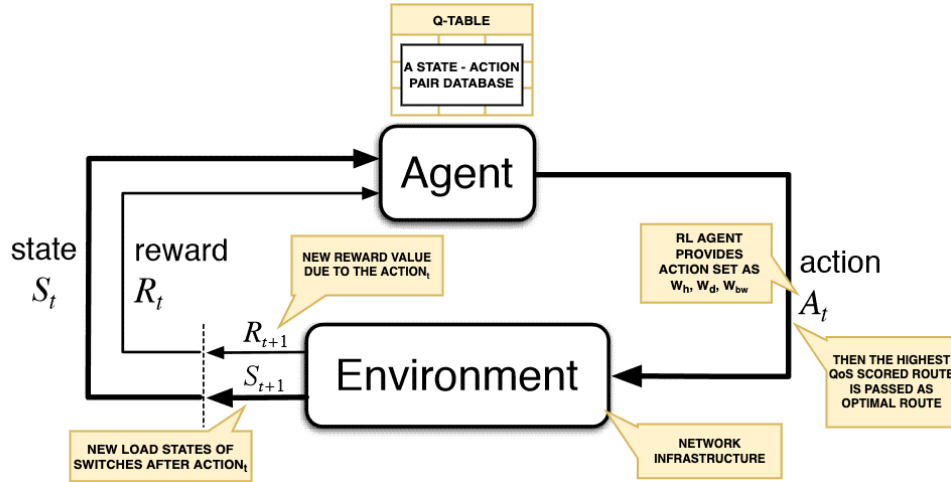


Figure 4.8 Proposed Q-learning based routing approach on computer networks domain (adopted from [2], and modified in networking domain)

4.3.2.2. Environment Hybrid network topologies, which are commonly used hierarchical and mesh, have access switches and middle and core layer switches in layers, respectively. As shown in Figure 4.9, hosts are connected to the network with access switches and forwarding is provided by middle and core layer switches mainly to which access switches are connected.

Middle and core layer switches undertake the main forwarding task and data load of the network. In RL context, data layer elements are defined as the environment in order to create a network management that pay attention to the load balance by obtaining the load status on the switches. The network topology communicates with the system bidirectionally and provides this communication with OpenFlow. This protocol makes possible to achieve instantaneous status of the network elements to the control layer, while issuing new flow actions from control layer.

4.3.2.3. State Q-learning is the algorithm that tries to apply the most ideal action according to the state of the environment. Network Statistic Tracker obtains periodic information from the network environment, and the Switch Load Tracker determines the percent load of the switches L_{sw} using (1). In order to narrow the load status space of the switches in the network environment, the load percentages between 0 and 100 are defined in

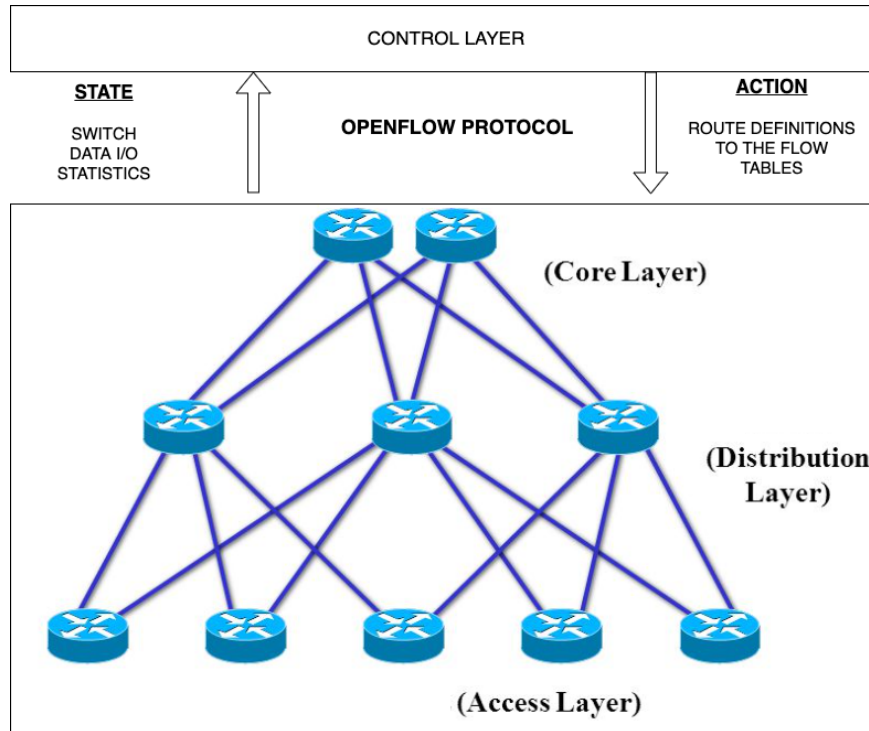


Figure 4.9 An illustration of mesh-hierarchical topology definition as Data Layer with OpenFlow messages (taken from [6])

4 quarters. Switches whose load percentage is calculated are assigned to one of four different positions as LP_{sw} with the help of Table 4.3.

Table 4.3 Conversion table of continuous load percentage to the discrete position for switches

Discrete Switch Load Positions LP_{sw}	Actual L_{sw} Ranges
0	0% - 25%
1	25% - 50%
2	50% - 75%
3	75% - 100%

Using the Table 4.3, switches whose load state is measured in continuous range are represented discretely. In this way, the Q-learning search space is narrowed. Each switch can be in one of four different load state at any given time. For a simplified Q-learning

implementation, it is necessary to determine the total number of discrete states in which the network environment can expose. Thus, the network state at any moment can be represented by s_i . In order to achieve this, the exponential product of total number of selected switches M with the total number of positions p gives the total number of discrete states as S of the system. Additionally, the S value is used to define Q-table for representing the each unique system state.

$$S = p^M \quad (6)$$

Eq. (6) gives the total number of discrete load states of the network environment. All the load combinations the system can fall within this amount of variation, and the system is in a state in this range at any given time.

$$s_t = \sum_{j=1}^M LP_{sw_j} \times p^{(M-j)} \quad (7)$$

To find actual state s_t of the network environment at time t , eq. (7) is used. The load status tracked switches are located in a position between 0 and 3. Then eq. (7) is used to express all positions of the network environment with a single state number s_t . This formula is used both to determine the action at production phase or to update the Q-table when training phase.

4.3.2.4. Action Q-learning selects the most appropriate action from an action set A to balance the system against actual load status of switches. The predicted action a consists of W_h , W_{bw} and W_d . Since the eq. (5) is a weighted sum of QoS ratios, the summation of the action weights have to be 1. Additionally, the continuous search space of weight values is narrowed by defining these on a discrete range. For this purpose, Table 4.4 is defined to make action selection. Each a_i is a triple combination of the weights of the QoS factors. We created 21 different actions by representing the coefficients that can take values between 0 and 1 in multiples of 0.2.

Table 4.4 Action numbers and their weight equivalents in discrete range

Action Numbers	(W_h, W_{bu}, W_d)
0	(0.0, 0.0, 1.0)
1	(0.0, 0.2, 0.8)
2	(0.0, 0.4, 0.6)
3	(0.0, 0.6, 0.4)
4	(0.0, 0.8, 0.2)
5	(0.0, 1.0, 0.0)
6	(0.2, 0.0, 0.8)
7	(0.2, 0.2, 0.6)
8	(0.2, 0.4, 0.4)
9	(0.2, 0.6, 0.2)
10	(0.2, 0.8, 0.0)
11	(0.4, 0.0, 0.6)
12	(0.4, 0.2, 0.4)
13	(0.4, 0.4, 0.2)
14	(0.4, 0.6, 0.0)
15	(0.6, 0.0, 0.4)
16	(0.6, 0.2, 0.2)
17	(0.6, 0.4, 0.0)
18	(0.8, 0.0, 0.2)
19	(0.8, 0.2, 0.0)
20	(1.0, 0.0, 0.0)

RL agent chooses an action either randomly or using Q-table which consist all states of the system and their Q-values for any action in training phase. When the agent chooses an action from Q-table, it selects the highest Q-value for the network state to pass appropriate action to the system.

4.3.2.5. Reward The controller makes changes on the load positions of the switches after flow routing. An unbalanced increase of load positions on switches can result in congestion, packet delay, and packet loss. For this reason, a mechanism that rewards the approach that equally distributes the load passing over the switches at any given moment is ideal. The reward function we propose provides a reward that is inversely proportional to the size of the load situation. This encourages the lower load positions for each switch and imposes a penalty in the case of a higher load. On the other hand, situations may occur where more than half of the switches exceed a certain load density level when exposed to heavy traffic

in network elements. In such peak times, a decreased penalty system has been designed to provide stability . The proposed reward algorithm is shown in Algorithm 2.

Algorithm 2 Reward Algorithm

Require: P is a HashMap that contain switch-load position pairs as (sw_i, p_i)

Require: M is the total number of selected switches

$ZeroPosSwCount \leftarrow Counter(p_i = 0 \in P)$. \triangleright $Counter$ counts the HashMap values

$OnePosSwCount \leftarrow Counter(p_i = 1 \in P)$

$TwoPosSwCount \leftarrow Counter(p_i = 2 \in P)$

$ThreePosSwCount \leftarrow Counter(p_i = 3 \in P)$

if $TwoPosSwCount + ThreePosSwCount = 0$ **then**

$R \leftarrow 2.5 \times ZeroPosSwCount + 1.5 \times OnePosSwCount$

else

if $(TwoPosSwCount + ThreePosSwCount) \geq M/2$ **then**

$R \leftarrow -(2.5 \times ZeroPosSwCount + 1.5 \times OnePosSwCount)$

else

$R \leftarrow -(2.5 \times ThreePosSwCount + 1.5 \times TwoPosSwCount)$

The reward algorithm has a mechanism that decreases the penalty as the load rate of the switches increase, in cases where more than half of the switches in the network environment are at 2 and 3 load positions. In the other case, it has a motivation that rewards the RL agent more for the less load status. This approach encourages the RL agent to form a network environment that is in a load balanced status in any time.

4.3.2.6. The Optimal Policy Our proposed policy is designed to maximize the reward in the Q-learning routing process. In this way, while choosing the action a , RL agent learns the routing with a high QoS score (lower hop count, higher bandwidth, and lower delay) for flow requests and to balance the system. In the training phase episodes RL agent converges to the optimal Q-function by trying all action-state pairs in (8). Episode refers to the *flow* in networking domain. In training phase, many flows are generated in a traffic scenario, and pushed to the algorithm to obtain a well-trained Q-table at the end of the training. It updates the Q-values in the Q-table. It determines the action that balances the load distribution of the system and provides the highest QoS for the flow requests. The Q-value in the Q-table is

used to find the best action. The Q-value is a measure of the overall expected reward when the RL agent is in the s_t and performs the a_t at time t .

Q-table is a matrix within the scope of the RL approach which contains Q-values for all action-state pairs for the environment. In a well-trained Q-table, the a with the greatest Q-value for each s represents the best a that can be taken in that system s .

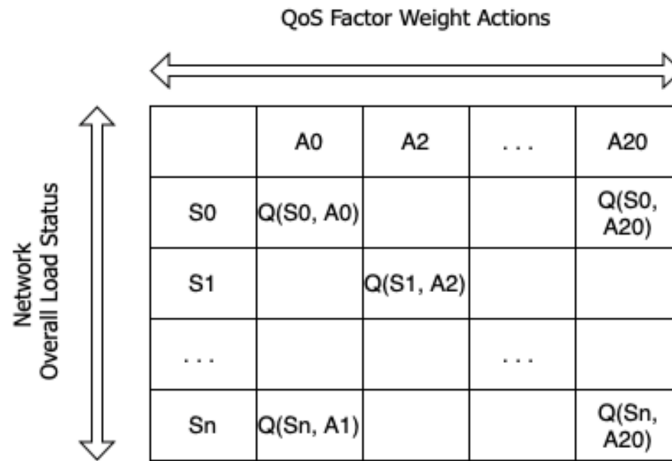


Figure 4.10 Q-table visualization that shows Q-values by index of actions and states

The illustration of the Q-table is shown in below Figure 6.2. Each row represents a network state s , and on the other hand, each column represents an action $a \in A$. Since the number of all load states of the system is S , it is equal to the total number of rows. The total number of columns is equals to 21 which is the number of combinations of weight factors A as shown in Table 4.4.

$$Q_{t+1}(s_t, a_t) = (1 - lr) \times Q_t(s_t, a_t) + lr \times [R_t + \gamma \times \max_a(Q_t(s_{t+1}, a)) - Q_t(s_t, a_t)] \quad (8)$$

The RL agent updates the Q-values in the Q-table with eq. (8) in order to obtain the most optimal Q-table during the training phase. In the equation, lr is the learning rate that the algorithm have to make in each iteration and have to be in the $[0,1]$ and it determines the

weight of newly acquired knowledge compared to the previous one. γ is the discount factor used to balance the immediate and future rewards. R_t is the reward that acquired for the a_t at time t . These hyperparameters are used to tune the converging speed of the algorithm. The terms between square brackets show the updated value that is the difference between the current estimate of the optimal Q-function $Q_t(s_t, a_t)$ and new discounted estimate. The new $Q_{t+1}(s_t, a_t)$ depends on the $Q_t(s_t, a_t)$, s_t , a_t , R_t , s_{t+1} .

4.3.2.7. Exploration and Exploitation Technique In the training phase, Q-learning offers two ways to determine the best action for the acquired system state. There is a balance between choosing the optimal expected action (exploitation) and choosing a random action (exploration) in the hope that it can bring a greater reward in the future. At this point, an ϵ -greedy method with linear decay value ld for exploration and exploitation is defined as in eq. (9) The logic of ld is defined in Alg. 3. In each episode on training, the agent explores if the random value is greater then ϵ else exploits. ld is defined to make the algorithm more exploit-oriented after each episodes. The ld is calculated as difference of the highest and lowest epsilon values is divided by the total number of episodes. Then each episode, it is subtracted from the ϵ as shown in below.

$$a_t = \begin{cases} \max_a(Q_t(s_{t+1}, a)) & \text{if } x < \epsilon - ld \\ \text{random}(A) & \text{otherwise} \end{cases} \quad (9)$$

4.3.2.8. Training the Q-learning model This section discusses the training procedure of RL for obtaining the Q-table to be used in the routing process. Before the tests we run in this thesis, it was aimed to develop a model that balances the load of the network elements and provide optimal QoS for each flow by training the proposed model for various network scenarios.

The routing algorithm in Alg. 3 implements the learning process for our RL-based approach to achieve a well-trained Q-table. In recent work [55], a similar Q-learning training procedure

Algorithm 3 Q-learning Training Algorithm

Require: lr is learning rate
Require: γ is discount factor
Require: $EpisodeCount$ is the total number of flows
Require: ϵ is the exploration and exploitation parameter
Require: ϵ_{min} is the lowest value of the ϵ
Require: F is an array that contains as f (src-ip, dst-ip) pair of flow requests
Ensure: $length(F) = EpisodeCount$
 $ld \leftarrow (\epsilon - \epsilon_{min}) / EpisodeCount$ $\triangleright ld$ is the linear decay value
Initialize Q-table as 0-matrix with shape of $(S, A), \forall s \in S, \forall a \in A$
while $F \neq \emptyset$ **do**
 $f \leftarrow F.top()$ $\triangleright f$ is the first (src-ip, dst-ip) pair of the F and it is popped
 acquire s_t using Eq. 7
 $\epsilon \leftarrow \epsilon - ld.$ \triangleright linear decay is implemented
 select a_t for s_t with policy derived using Eq. 9
 get available *routes* on the f using the Sec. 4.1.
 get QoS metrics $(\hat{h}_{route}, \hat{d}_{route}, \hat{bw}_{route})$ using the Sec. 4.2.
 while *routes* $\neq \emptyset$ **do**
 $QoSScore \leftarrow$ with QoS metrics and a_t as (W_h, W_d, W_{bw}) using Eq. 5
 $QoSScores.append(QoSscore)$
 routes.pop(0)
 select *route* that has maximum the $QoSScore$
 update the flow tables of related switches for the *route* using Sec. 4.4.
 acquire s_{t+1} using Eq. 7
 get R_{t+1} for s_{t+1} using Alg. 2
 update Q-table for $Q_{t+1}(s_t, a_t)$ value using Eq. 8
 store the trained Q-table to use in test scenarios

is used. Their routing algorithm implements the learning process used to find the best paths for all the pairs of nodes on the data plane. But our design in our Alg. 3, we store Q-values in the Q-table which are determined by the according to the network load status. When we go back to the algorithm, in learning process, the algorithm receives lr, γ, ϵ parameters to utilize in Q-function. Also, ϵ_{min} is provided to calculate ld values for decaying ϵ in each iteration. In training phase, we provide some large scale traffic scenarios to enable a comprehensive learning for traffic patterns with array of flows as F .

The objective of the learning process is the gathering best Q-table to make it possible QoS and load-aware routing in production. At the beginning a Q-table is initialized with the shape as (S, A) . When each flow is requested in a loop, RL-agent acquires the actual s_t of

the environment using eq. (7). To define an action, algorithm generates a random number between $[0,1]$ and selects an a_t according to the exploration and exploitation policy in eq. (9). Further, available *routes* are found using the Sec. 4.1. modules, and then $QoSScore$ is calculated for each route with eq. (7) to identify most ideal *route*. Later on, controller initializes the *route* on the flow tables of the switches and acquires s_{t+1} to obtain new load status of the network environment. At this point, the reward or penalty is obtained for the new state from the reward algorithm in eq. 2. Subsequently, the Q-table is updated with eq. (8), which takes into account the new state of the system and the discounted maximum value for the future state. In this way, the Q-table is updated with each new flow request and the action set that performs the load balancing for the overall environment.

After all, RL-agent completes the training procedure and achieves the updated Q-table. The resulting Q-table has the ideal state-action pairs to provides best QoS weights that makes the system more load-balanced and QoS-assigned for each flow requests.

4.4. Flow Starter Module

After selecting the best route using one of the Section 4.3. routing methods, this module updates the data layer elements. It makes this by updating the network switch's flow tables which exists in the route path. The flow tables are updated on the flow tables of the switches with the OpenFlow messages that contain the forwarding information according to the selected route. In this way, the controller completes the flow initialization process for a flow request.

5. EXPERIMENTS AND RESULTS

In this section, we discussed the preparation processes of our experiments and then we analysed the obtained test results. Firstly, the simulation environment is described for the networking tests in Section 5.1. Flow generation in the network environment and the traffic scenarios for the tests are detailed in Section 5.2. and 5.3. respectively. Before starting the tests, performance metrics are introduced in Section 5.4. Subsequently, the tuning methodology for the selection of learning parameters of proposed RL-based approach is examined in Section 5.5. Finally, the results and analyses we have obtained from our tests are discussed in Section 5.6.

5.1. Test Environment

The network topology preference of this study is similar to the well-used CAN topology that belongs to a defense industry company that is located in a large area campus. Because of the modular structure and easy expandability, hierarchical and meshed hybrid network topology is frequently preferred in CAN design [6].

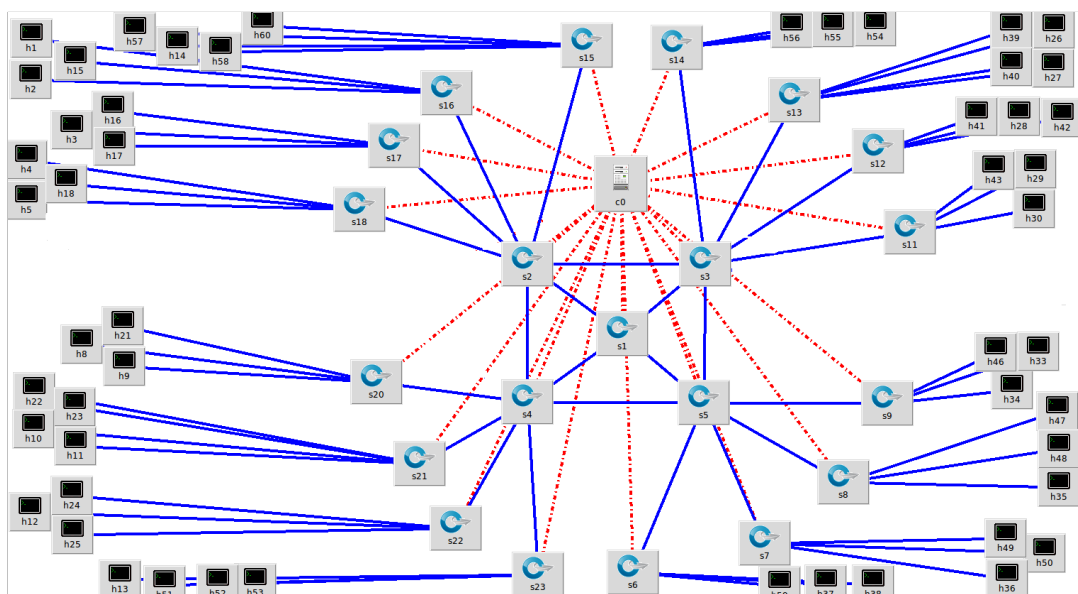


Figure 5.1 An illustration of the hierarchical and mesh topology design for the simulations

As seen in Figure 5.1, the core switch at the center of the topology, the distribution switches are at the middle level, and the access switches are located in the bottom level as an hierarchical approach. Also, the distribution and core levels are structured as mesh-like. Access switches are linked to any hosts which can be either client or server or both. The actual routing process of data traffic takes place at the distribution and core layers (see Figure 4.9).

The topology which is designed in the Mininet [66] environment consists of a total of 52 hosts and 21 switches. In addition, there are a total of 24 links between the switches. To more effectively test the capabilities of the routing approaches according to meet load balancing and QoS requirements throughout the tests, the bandwidth capacity of all links is defined as 20 Mbps and the link transmission delay is 1ms as default. Next, real-world IoT devices, manufacturing devices and user devices are simulated by connecting 3 or 4 clients to each switch. (See Appendix A)

In the tests we used traditional STP approach, the controller only collects data from the data layer via OpenFlow for analysis and switches perform routing and forwarding tasks themselves in accordance with TCP/IP. The switches are defined as OpenvSwitch and can operate as a traditional L3 switches in the STP approach. On the other hand, proposed SDN-based approaches perform routing task only on the controller.

5.2. Flow Generation

The network packet generator tool *iperf3* [67] is used to generate network traffic between pair of nodes for our Mininet-based network topology. *iperf3* allows to create UDP and TCP flows between client and server via simple scripts. It also make possible to tune of various flow parameters such as number of bytes or packets to transmit, time length of test, UDP bandwidth in seconds. Also, *iperf3* creates a report file for each flow generation and updates it for transmitted throughput, bandwidth rate, loss rate, delay and jitter in defined time interval as real time. We use these logs in the 4.1.2. to track network statistics and generate test results (See Appendix B).

5.3. Test Scenarios

In order to perform tests as close to reality as possible, we offer three different traffic scenarios consisting of elephant and mouse flows. When real internet traffic is observed, the minimum 60-80 percent of the traffic density consists of elephant flows. There are various definitions that commonly categorize flows as elephant and mice [68, 69]. In this study, flows which have greater than a certain size are defined as elephant, as stated in [70]. Remainders which are smaller than a certain size are defined as mice. Also, [70, 71] observed a high correlation between flow size and flow. For these reasons, in our study, in order to define a traffic consisting of elephant and mice flows, these are defined according to the flow rate and flow size pair. The flow rate and flow size selection sets are illustrated in Table 5.1

Table 5.1 Predefined flow size and rate sets for mice and elephant flows

Parameters	Settings
Flow sizes for elephant flows (in MB)	from 20 to 90, increasing by 5
Flow rates for elephant flows (in Kbps)	from 4096 to 14336, increasing by 2048
Flow sizes for mice flows (in MB)	from 1 to 10, increasing by 1
Flow rates for mice flows (in Kbps)	from 1280 to 3072, increasing by 256

In scenario installation, the flow queue for mouse and elephant flows is created by random sorting. After that, flow arriving times are generated with the help of Poisson distribution. Then, when the time arrives for any flow, two hosts are randomly selected from the network environment and a $(flowSize, flowRate)$ pair is selected according to the flow type using Table 5.1. Subsequently, the flow is initialized via *iperf3* using information $(SrcIP, DstIP, SrcPort, DstPort, flowSize, flowRate)$. In order to test each method used in the tests with the same scenario setup, each scenario is fixed with the seed factor (See Appendix A).

Three different scenarios have been defined to be tested within a certain period of time and same topology as detailed in 5.1. The percentage of flow types are diversified to change

network intensity. Since the starting time of the flows is independent of each other, the Poisson distribution is preferred [15]. Details of the test scenarios are given in Figure 5.2.

Table 5.2 Suggested traffic scenarios for tests

Characteristics	Scenario 1	Scenario 2	Scenario 3
Number of flows	100	200	300
Number of mice flows	30	70	105
Number of elephant flows	70	130	195
Elephant flow ratio	%30	%35	%35
Flow arriving time range with Poisson dist. (λ) sec.	8	6	4

We have developed and run all test in Ubuntu 20.04 (64-bit) Virtual Machine. The device configuration is 16GB RAM and 4 cores that are powered by Intel i5-10gen. Python version 3 and its libraries are preferred for development. Ryu controller with OpenFlow protocol is used to design the control layer and implement all APIs between data and application layer.

5.4. Performance Criteria

In the previous researches that we analysed and are given in Table 3.1, flow delay time, packet loss, jitter, throughput rate are used to measure the provided QoS performance for the flows. To provide same evaluation ground, we measure them in our tests. In addition to these QoS performance metrics, we compared the flow completion times for each approach. This gives an idea of the appropriate bandwidth choices of the routing methods. Additionally, we compared the total loads passing through the switches with cumulative intervals to measure the effects of the methods on the system load balance. Also, mean received packet per second is calculated for each approach.

To measure the delay, loss, jitter, bandwidth and packet per second of each flow, we use *iperf3* flow reports. When a flow initialized as determined in 5.3., *iperf3* creates a log file and updates it in given time of interval. To measure more precise performance results of each method, we prefer 1-second update period for any flow. The log document is prepared

as a listener on the destination host that namely server side. In each second, the amount of transferred data to the host, throughput rate, jitter, number of packets and losses, latency time are recorded. After the each test is done, the data in these reports are collected within themselves and the result graphs are obtained (See Appendix B).

Additionally, we measure the flow duration times to compare routing performance of tested methods. Because, route selections with higher available bandwidth are expected to complete the flow in less time. It is calculated using the flow log reports. Lastly, we propose a load balance comparison schema while providing time-based switch load monitor. With OpenFlow, we monitor the total amount of load passing over the switches at certain intervals of the test and in the end of the test.

5.5. Tuning the proposed RL-based Approach

ML algorithms require a training phase to achieve the desired level of optimized model based on data. The training data that is used in training phase can be either static or dynamic style which is streamed via simulators. The aim of training phase is to obtain the best model that solves the studied problem. In order to achieve this, various hyperparameters are used within the scope of the mathematics of the ML algorithm that is used. These hyperparameters are tuned on a performance criterion and time axis within the specified value range. The tuning procedure can be based on a searching randomly or in an algorithmic way [72]. In any case, the aim is to find the best values of the tested hyperparameters that converge fast and provides the highest performance. Within the scope of this study, lr and γ are the most effective hyperparameters on performance of the Q-learning algorithm are tuned and examined under different values to achieve the optimal model. Each tuning test is executed up to 5000 flows under same traffic settings. The ϵ value is decreased in between 0.75 and 0.1 range for each trial with a linear decay rate ld according to the total number of episodes to reflect the impact of exploration and exploitation. Therefore, the epsilon value is reduced by 0.00013 in each episode. The flows are ordered in a queue as randomly and flows are initialized in an arrival time row with Poisson distribution. The properties in the Table 5.1 are used to initialize

each flows and the traffic is created. The proportion of flow types are defined 1000 and 4000 for elephant and mice flows respectively. After the whole training and tuning procedure, obtained Q-table is integrated to the test phase to use in the test scenarios and analysis (see the Algorithm 3).

Reward-oriented RL model optimization in networking concepts is preferred previous works [56, 58] in this context. To examine converging performances of each trial, we compare the models according to the exponential moving average (EMA) of the rewards. Unlike the simple moving average or using only the reward value, EMA considers defined previous range of rewards of the episodes and gives more weighting or importance to recent observed data. Thus, while making more timely analysis, short-term fluctuations are reduced and the hyperparameters selection process is made with a weighted long-term trend analysis.

As seen in Figure 5.2, 5.3, 5.4 and 5.5, we construct a grid search in between γ and lr parameters. In total, 16 tuning tests are executed and each test simulation is run approximately 9 hours. In order to obtain best converged RL model, 4 different γ and lr values for each of them are tested. The tests are analysed as above by keeping the γ value constant and comparing it with a range of lr choices.

When we examine all figures, it is seen that when the γ decreases from 0.9 to 0.1, the amount of exponential rewards obtained by the tests in general increases. In figures 6.7 and 6.9, the γ choices of 0.3 and 0.9 and different lr selections did not cause much change on the tests which seem to follow similar patterns. In Figure 6.8, the lowest lr is very ineffective, on the other hand, other selections performing similarly. In Figure 6.6, it is observed that the test with the smallest selections in both parameters exhibits the best performance. Choosing the low valued γ and lr is interpreted as keeping the current status dominant and reducing the impact of future rewards while updating the Q-table (See the eq. 8). Considering the traffic on the network, short-term but effective action choices are made by RL agent to reduce the impact of instantaneous big changes (such as elephant flows with high bandwidth requirements) in order to keep the system in balance. According to the results, best parameters for RL model are 0.1 and 0.1 for γ and lr . With this selection, EMA

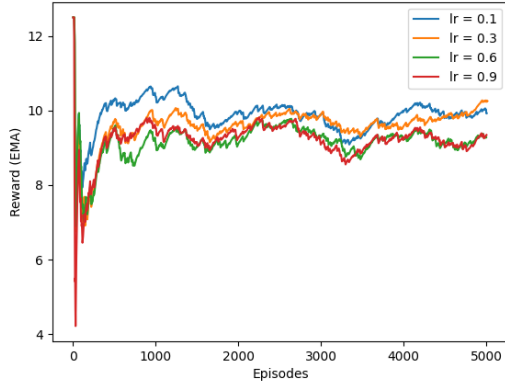


Figure 5.2 EMA of R for different lr , $\gamma = 0.1$

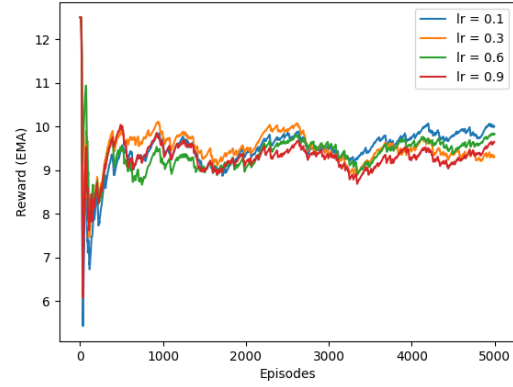


Figure 5.3 EMA of R for different lr , $\gamma = 0.3$

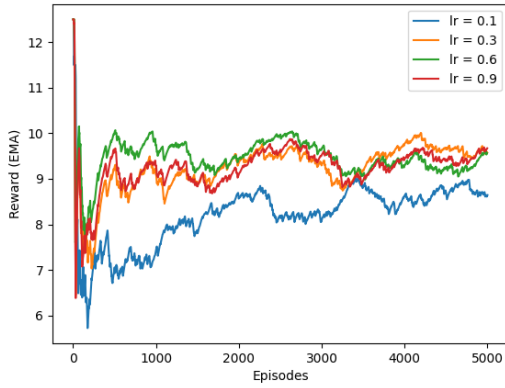


Figure 5.4 EMA of R for different lr , $\gamma = 0.6$

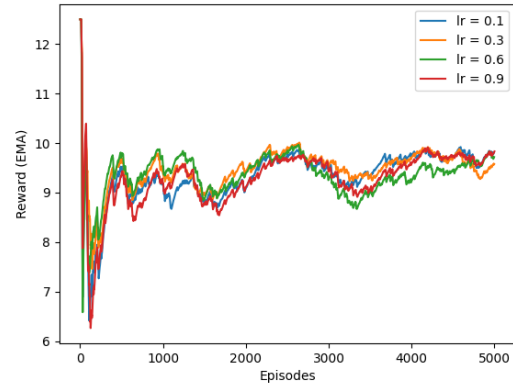


Figure 5.5 EMA of R for different lr , $\gamma = 0.9$

of reward is performed over any other test in the long run and we have saved its Q-table to use in the scenario tests as our proposed RL approach.

In order to evaluate the accuracy of the selected model as a result of fine tuning, additional analyses are made with the sliding window method using the cumulative moving average (CMA) of the reward (See Appendix C).

5.6. Results and Analysis

In this part, QoS and load balance performance tests are simulated under three network traffic scenarios for three different routing approaches. Traditional STP, Equally QoS-aware

Route Selector and Q-learning based Adaptive QoS and Load-aware Route Selector methods are determined as routing methods. Each routing method is simulated separately under 3 different traffic intensity described in 5.3. At the end of each scenario simulation, the methods are analysed for 7 different performance criteria which are detailed in 5.4. These performance criteria are analysed for QoS requirements of mean latency, total packet loss, mean jitter, mean bandwidth usage and mean number of packets per second analysis. Additionally, randomly selected flows are compared for each method according to their total duration times. At the end, the load distribution on the switches has been analysed in order to examine the network load allocation in the predefined network environment detailed in 5.1.

5.6.1. Bandwidth provisioning capacity

In bandwidth test, we have used *iperf3* flow reports to obtain mean received number of bits per second for each flow. Later on, the average bandwidth values of each flow which run under each scenario are summed and divided by the total number of flows that belong to the scenario. In this way, the capacity of routing methods to provide average bandwidth per flow is found for each scenario.

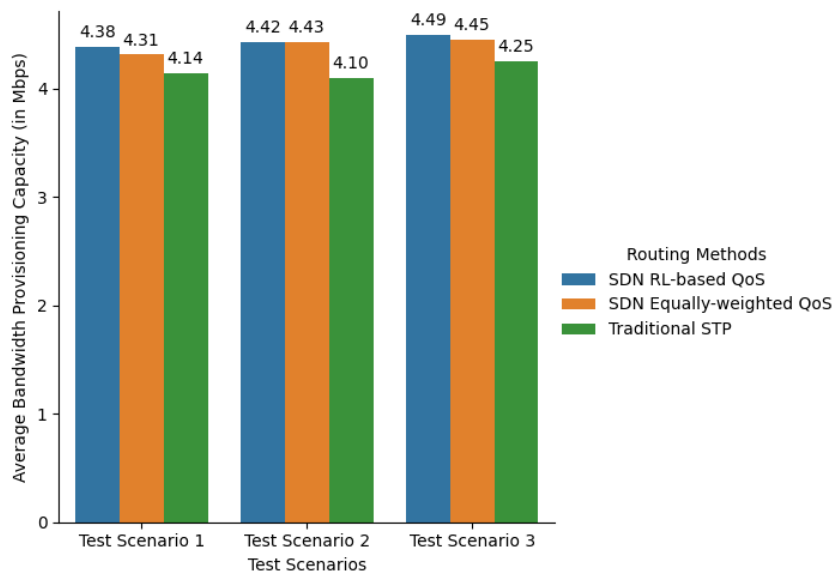


Figure 5.6 Bandwidth provisioning capacities of the routing approaches

In Figure 5.6, the average bandwidth provisioning capacities of the routing algorithms for each scenario are evaluated. In all tests, all methods transmitted over 4 Mbps of data to the target devices. The proposed RL-based routing approach provided approximately 4.5 Mbps of bandwidth in Scenario 1 and Scenario 3 and used system resources more efficiently than other methods. In Scenario 2, our method has almost equal service providing result with the equally weighted QoS routing approach.

5.6.2. Average delay time

In this test, the average delay time in milliseconds per flow of each scenario is analysed. *iperf3* flow report provides average delay time for each flow at the end of the each flow. To obtain an average delay time for all the tests which are run around each scenario, we summed up the all average delay times and divided to the total number of flows.

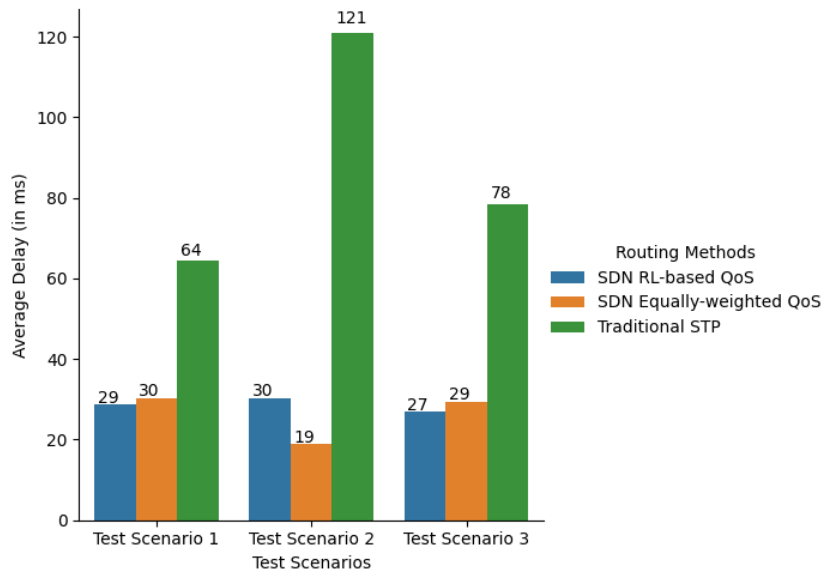


Figure 5.7 Average delay time results of routing algorithms according to scenarios

The programmability effect of the SDN-based approaches in latency management is directly seen in Figure 5.7. The routing choices of STP from fixed routes by blocking some switches while routing caused high delays. Our proposed routing approach performed effectively in Scenario 1 and Scenario 3, but remained around 30 ms in Scenario 2.

5.6.3. Average jitter time

This test handle the average jitter time for flows for each scenarios. The *iperf3* tool reports the occurred jitter in a second period for each flow. We found the total jitter time for each flow and divided by the duration of it to find and average jitter time per flow. Afterwards, this process is averaged for all flows within the scope of the scenario.

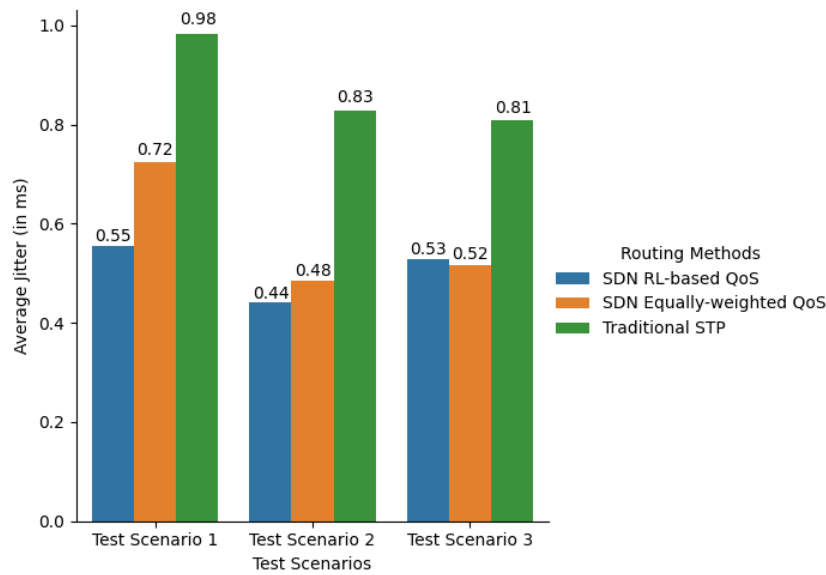


Figure 5.8 Average jitter time results of routing algorithms according to scenarios

It is an important point that the jitter performance of smart routing methods is better than traditional methods. Because, while writing to the switches' tables by control unit in SDN, network state information is transmitted to the control layer simultaneously. In the situation, a poorly designed algorithmic approach can bring additional delay and standard deviation of the delay. However, the proposed RL-based routing approach performed jitter below 0.6 ms on average in all tests, while it is slightly behind the other method only in Scenario 3.

5.6.4. Total packet loss

This analysis gives information about the number of packets dropped due to the route preferences of the tested routing methods that are run on the scenarios. The total number

of dropped packets for all flows are summed under the scenarios and compared with respect to routing methods.

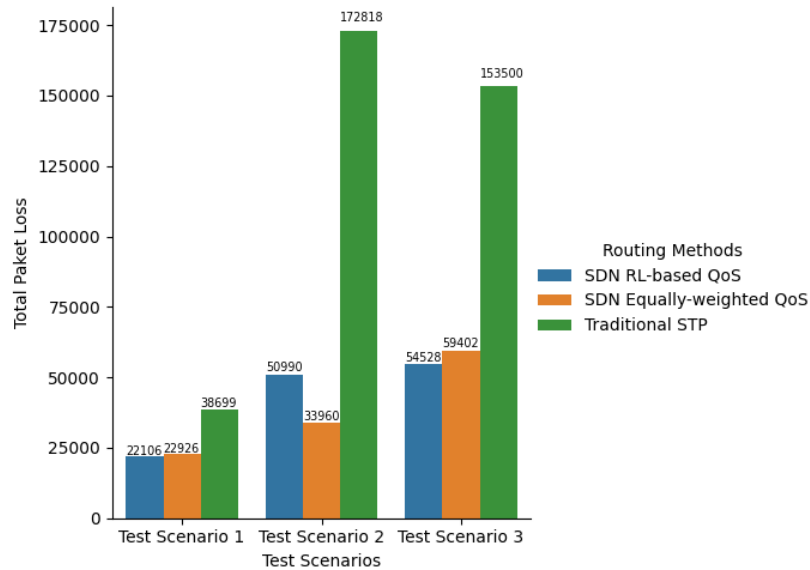


Figure 5.9 Total packet loss results of routing algorithms according to scenarios

RL is a ML approach that seek the best solution space when it focuses to the long-term rewards. In this context, it is important that the proposed approach kept packet loss to a minimum in Scenario 1, which is a relatively short-term test as seen in Figure 5.9. Additionally, our model remained below for packet loss under the 55000 in total in Scenario 3 which is the longest test scenario than others. When the trends of packet loss results of the methods are examined, our approach has a decreasing packet loss rate.

5.6.5. Average received packet per second

Average received packet per second analysis is another performance measure metric of the QoS. The higher this value is within the scope of the capacity of the network components, the better the system capacity utilization. The average amount of packets transferred per second for each flow is added up and divided by the total number of flows. In this way, a knowledge about the instant packet transfer capacity of each routing method is obtained.

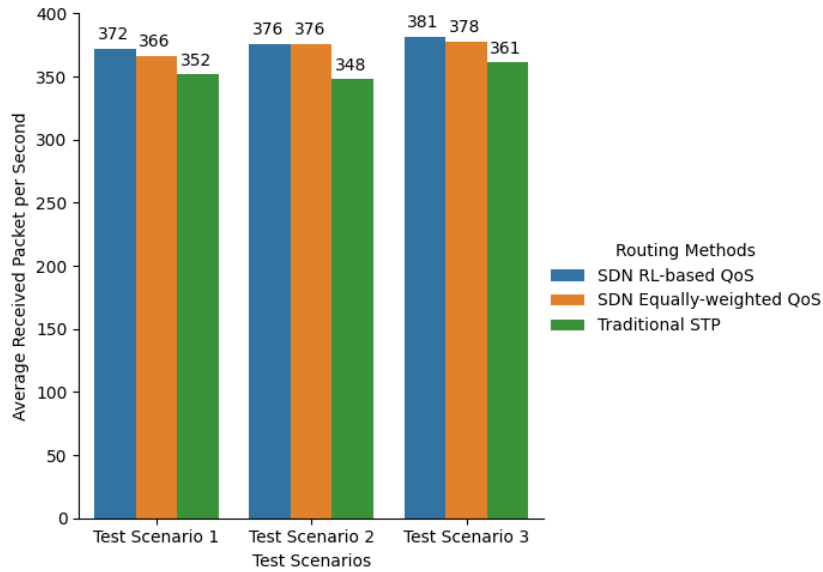


Figure 5.10 Average received packet per second results of routing algorithms according to scenarios

The average number of received packets gives important clues for network capacity utilization. It is normal for each of the routing methods to be close at this point, as the routing operations are done somehow. While all of the methods successfully transmitted around 340 or more packets in average to the target hosts, our approach had the best performance in Scenario 1 and Scenario 3, but equal performance in Scenario 2 as seen in Figure 5.10

5.6.6. Flow duration

Flow time analysis aims to observe the provided bandwidth by the tested routing methods for selected flows. Since the traffic generated in the scenarios is exactly the same, the starting times of the examined flows are same, but the ending times may vary according to the provided bandwidth capacity of the methods.

In this section, flow time analysis has been calculated for each scenario. The total duration of the randomly selected flows from each test is examined in terms of routing methods. The randomly selected flows for analysis within the scenarios are exactly the same setup for each routing method.

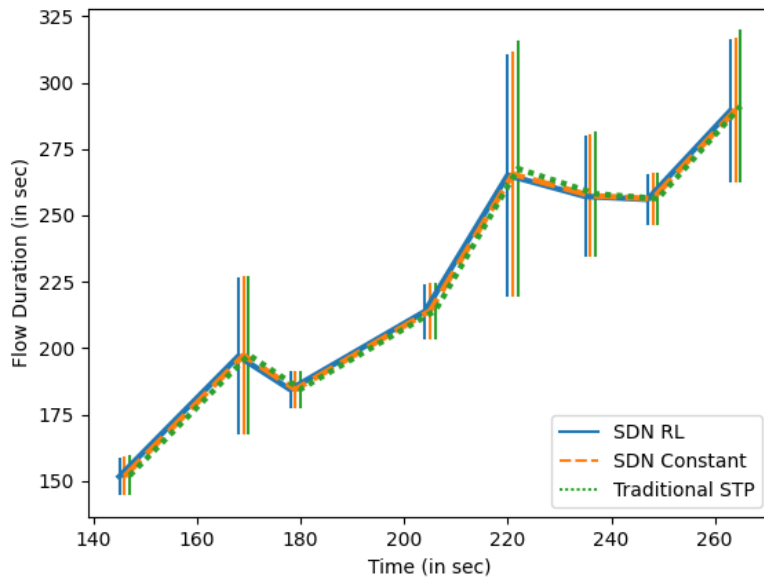


Figure 5.11 Flow duration analysis for Scenario 1

As seen in Figure 5.11, eight flows are randomly selected under Scenario 1. The lines parallel to the y-axis of the graphs give the duration time of each routing technique for the relevant flow. When examined in this context, the routing method we propose is always completed in a shorter time, especially in long-term flows.

When the results obtained in Figure 5.12 are examined, a total of 11 randomly selected flows are discussed. STP method is worst, especially in long-term flows, when our proposed method completed the flows in shorter times compared to the equally weighted QoS method.

In Figure 5.13, we examined the duration time of randomly selected 15 flows in total, which are obtained through Scenario 3. As seen in figure, this scenario has a longer and more intense traffic than the other tests, however our approach seems to perform much shorter times, especially in elephant-like flows.

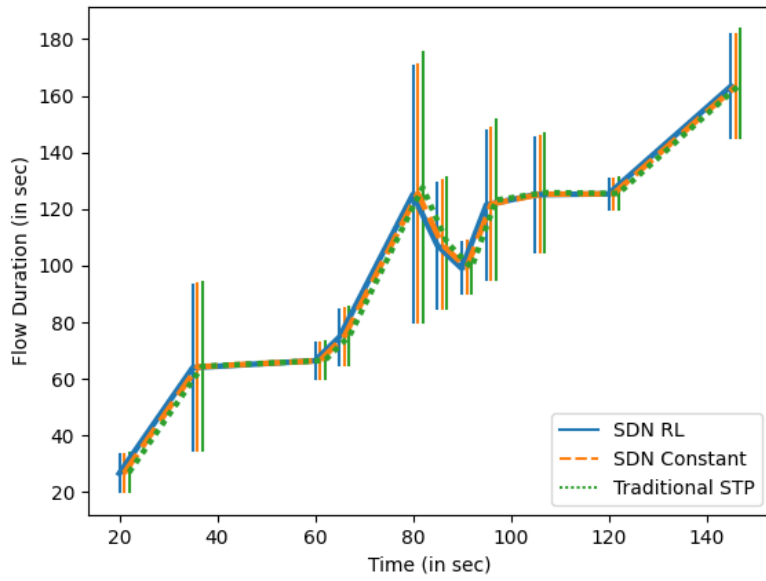


Figure 5.12 Flow duration analysis for Scenario 2

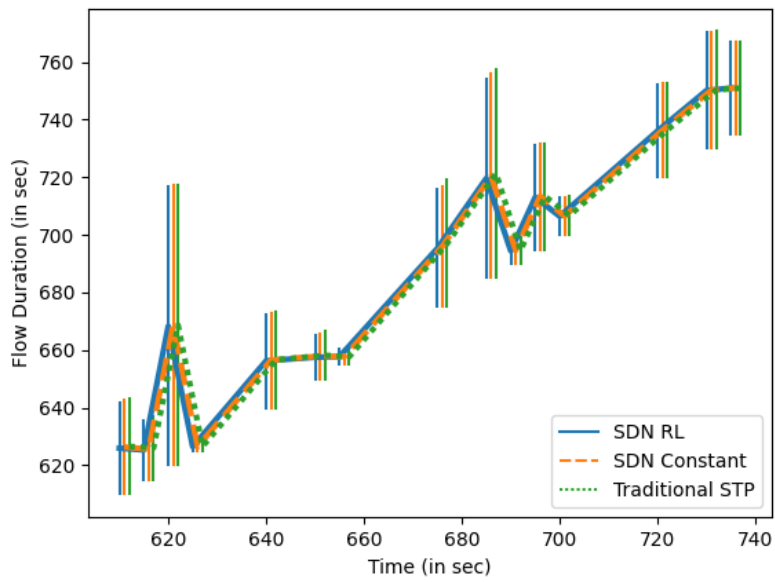


Figure 5.13 Flow duration analysis for Scenario 3

5.6.7. System load balance

System load balance test examines the cumulative load passing over network devices at certain time intervals. Many QoS issues on the network occur in the form of unbalanced use of network resources and then packets delays and drops. The transferred data through five switches on the distribution and core layers which are defined in the predefined topology is examined in the time axis for each routing approach.

The analyses is ordered as our proposed routing method, equally-weighted QoS routing method and STP method. Because of the intense and longer traffic characteristics, Scenario 3 is studied in the load balance analysis on the switches. In the Scenario 3 test, which is 1650 seconds in total, the cumulative amount of data passing through the relevant switches every 550 seconds is recorded. In this way, the usage intensities of the switches are analysed both in the intermediate times and in the long run.

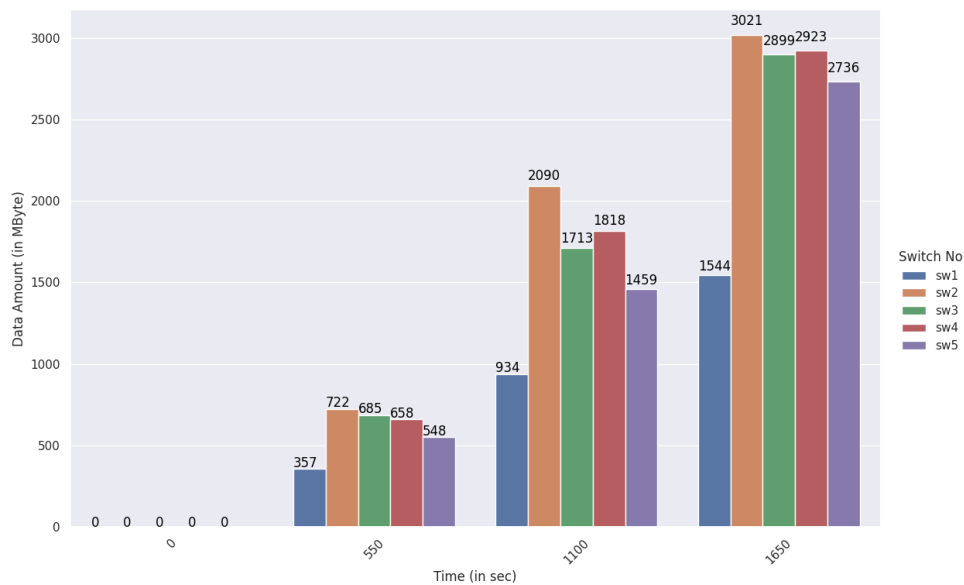


Figure 5.14 Cumulative received data amount on core and distribution switches for our proposed routing method

In Figure 5.14, the cumulative load distribution graph of our proposed routing method is shown in three discrete time periods. Except for Switch 1, the amount of cumulative data passing through other switches is close. In particular, it is expected that the an RL gives

better results in a longer period. However, when Figure 5.15 and Figure 5.16 are examined, the amount of data forwarded using Switch 1 is greater in our recommended method than other methods.

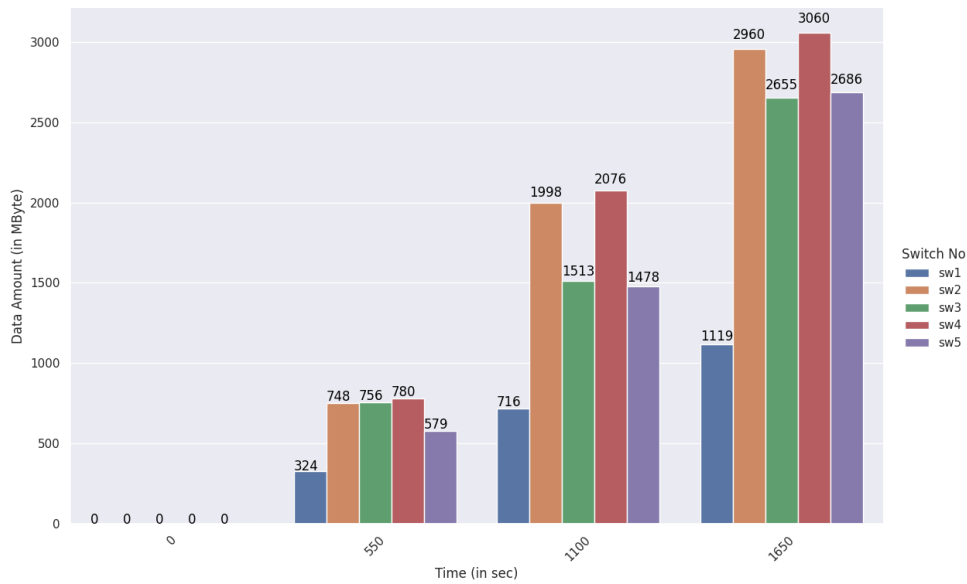


Figure 5.15 Cumulative received data amount on core and distribution switches for equally weighted QoS routing method

In Figure 5.15, when the loads passing through the switches in the time axis are examined, it can be said that a load balance is exist except Switch 1. However, a equally weighted QoS routing logic for QoS factors caused a load sharing bias between Switches 3-5 and Switches 2-4. When the reason for this situation is examined, it is seen that 27 hosts are connected to Switch 2-4 in the distribution layer and 25 hosts are connected to Switch 1-5 in total (See Figure 5.1). Thus this method, instead of distributing the data load to other switches, it has made a simple routing.

Due to the algorithm of the STP method, some switches are blocked to prevent loops and it causes in making load balancing efforts difficult. As seen in Figure 5.16, the effect of this blocking and free paths is noticeable on the switches negatively even in different time periods.

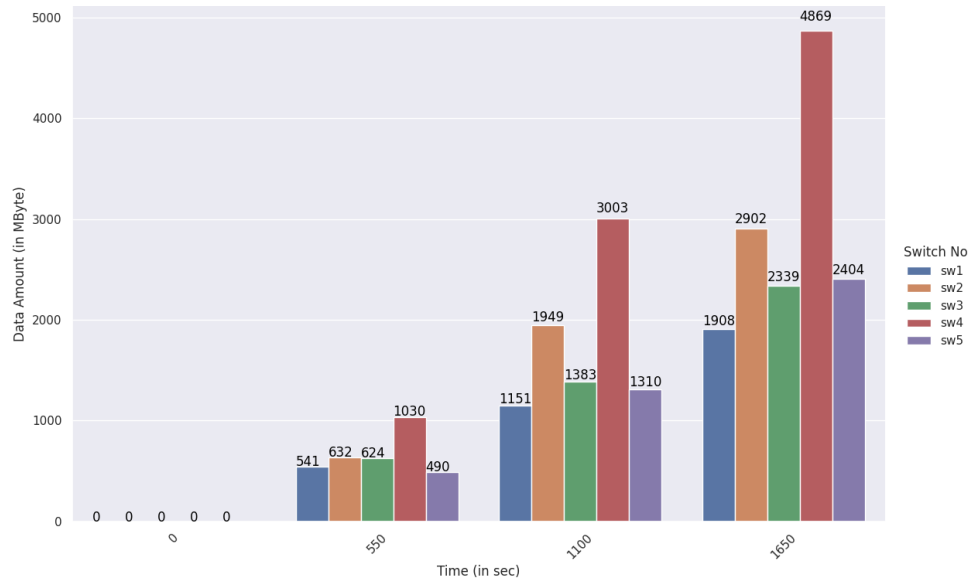


Figure 5.16 Cumulative received data amount on core and distribution switches for STP routing method

Table 5.3 Statistical evaluation of loads on the time intervals for each routing methods

Routing Methods	Time Period 1(550. sec)			Time Period 2(1100. sec)			Time Period 3(1650. sec)		
	SD	Min-Max Diff.	Median	SD	Min-Max Diff.	Median	SD	Min-Max Diff.	Median
SDN RL-based QoS	147	365	658	436.9	1156	1713	612.7	1477	2899
SDN Equally-weighted	192	456	756	542	1360	1513	789.2	1941	2686
Traditional STP	213	540	624	757.7	1852	1383	1164.1	2961	2404

In Table 5.3, the time period-based standard deviations of the loads passed on the switches are found. Based on the fact that the smallest deviation value describes the closeness of the variables to each other, our proposed approach obtained the smallest deviation values in all time intervals. Also, to additional information about the data load ranges, the difference of maximum and minimum loads and median values are provided for each time period. According to the table, our method has the minimum standard deviation results for all time periods. Additionally, the minimum ranges on the difference of minimum and maximum load values are achieved in the proposed routing method.

When we evaluate our intelligent routing approach and other SDN-based and traditional STP

methods, our proposal showed its advantages in various aspects in all analyses. Applying the load balancing for network sustainability and QoS requirements at the same time require being good at many performance metrics. For this reason, our method achieves effective results in all of them but it is open to development getting better results.

6. CONCLUSION

Today, the increase in the amount of data produced day by day and the variety of devices that generate the data is a warning for the network infrastructures. It is insufficient to manage computer networks, which are an extremely dynamic and variable environment, only by simple routing techniques such as best-effort. A network management that is unaware of the load status of the network and a neglected QoS needs of the flows dissatisfies the users and reveals various system problems. In this context, load balancing and QoS are among the most important topics of computer networks in terms of both QoE and sustainable usage of the network system resources. These issues have been studied for a long time with traditional routing methods and new generation programmable and centralized management approaches such as SDN [31–35]. While some of these studies focus on these issues separately [16, 44], and some of them focus on both simultaneously with inserting one of the QoS factors [15, 57].

Egilmez et al. [13] try to optimize the QoS requirements of multimedia flows and provide a flow classification method. After classification, multimedia flows are routed through QoS-guaranteed routes, while other flows are routed using a traditional method. In another flow classification method [14], flows are routed through bandwidth-guaranteed routes or multiple routes according to their type. Kosugiyama et al. [17] proposed a flow aggregation based routing approach to reduce end-to-end latency. Flow classification-centered studies put lowered QoS priority flows into the second plan. On the other hand, techniques that assign guaranteed routes for elephant flows can get congestion in a biased flow trend. Classifying the flows and sending them from certain routes or routing them in a aggregated way pose a threat to the load balancing of the network elements.

Shi et al. [15] suggest different routing policies for elephant and mouse flows. Nkosi et al. [16] propose routing to the new incoming flow requests from the route with the least occupancy rate. Network traffic has a dynamic structure. Therefore, routing according to the flow type as in the proposed studies or routing the flow on most appropriate available

bandwidth encounter problems in an excessively biased flow traffic. In addition, since only caring bandwidth is considered among the QoS factors, other factors such as latency and packet loss can not meet the requirements.

When the existing studies are examined, we have not come across any study that considers load balancing and QoS demands for the data layer at the same time, to the best of our knowledge. For this purpose, the first of our research questions is to develop a method that considers these two important concepts at the same time. In order to develop such a method, a structure that monitors both the load status of the data layer devices and the QoS factors is required. Considering the dynamic and variable nature of the network environment, there is a need for a method that can work well and fast in stochastic situations. In the work [44], which makes a weighted QoS calculation to make the most ideal routing, requires a long computation time for the simulated annealing method to give good results. In another ML supported work [49] clustering techniques have been used to find the most appropriate route, but the selected clustering methods cannot work successfully in large search spaces such as network environment. In this context, RL was preferred in our study as a technique that can take adaptive routing decisions in a dynamic network environment and does not have much computational load.

This study proposes an intelligent and effective routing method using the SDN paradigm. A Q-learning based load balancing and QoS-aware SDN approach uses RL to learn the network environment variables and determines optimal routes even at various traffic densities. Proposed approach tries to balance the data load passing through the network devices for the sustainability of the network environment and for each client to achieve sufficient QoS. It performs adaptive responding with Q-learning to provide more effective QoS in a democratized load distribution on the network environment simultaneously. The load balancing is performed by creating incoming routing requests with the most accurate QoS weights. The proposed approach periodically monitors the total delay, available bandwidth, and the total hop count of each flow request. When a flow request arrives to the controller, the QoS scores of the all suitable routes are calculated with the ideal weights for ranking and the highest scored route is selected for routing and then the routing is executed.

Proposed SDN controller design is compared with an equally weighted OoS-aware SDN routing approach and a traditional STP routing method. 3 different network scenarios are generated to evaluate routing methods' performances according to the QoS and load-balance requirements. Traffic scenarios are created with varying numbers of flows and generating flows at shorter intervals. All methods are evaluated in terms of latency, jitter, packet loss, number of packets per second and bandwidth capacity, which are critical for QoS requests. Additionally, we select randomly some of the flows and their total durations are compared for routing methods under different scenarios to obtain some insights about ability of QoS provision of each method. In order to measure the load balancing performance of the methods, cumulative load distribution of the switches are evaluated in short and long time periods.

All of the tests are taken under 3 different scenarios, our study achieved a relatively better result in most of the experiments. In the bandwidth provisioning capacity test, we compared how much data the routing approaches transferred on average per second across the traffic scenarios. This test also gives insight about how efficiently the network environment is used. Our proposed approach performed a data transfer over 4 Mbps in all tests, and achieved best performance in the first and last scenarios. In scenario 2, we obtain almost equal bandwidth providing result with the Equally-weighted QoS routing approach. In average delay time per flow test is inquired across routing methods. Our method results happened around 25 ms delay per flow on Scenario 1 and 3 as best performances. Only in the second test scenario there was a small increase in result and took the second place. The jitter is an important QoS factor for some applications. SDN-based routing approaches bring an additional data transfer overhead on the network. When the instantaneous state of the network is obtained from the data layer, communication with application layer takes place simultaneously. In such a case, a poorly designed control layer causes congestion problems such as delay and jitter can occur in the routing tasks. Our approach produced significantly lower average jitter in the first two scenarios. It also outperformed traditional STP, staying under 0.6 ms in all scenarios. Packet loss is a QoS factor that have to be considered especially for flow requests that need to be routed with the UDP. The minimum packet loss is preferred for a desired QoE. It is also

an indication of the load-balanced use of the network environment by avoiding congestion issues. Unlike other routing methods, the proposed routing technique has resulted in packet loss with a decreasing acceleration in the long run. As it is known, RL tends to choose the best actions by focusing on the long-term reward. In the first and last scenarios, it achieved around 22000 and 54000 packets in total packet loss, respectively. Average received packet per second test is suggested as a way of examining the average bandwidth provisioning test in packet size. The proposed method completed the test results in the first place, except for the Scenario 2. Parallel to the intensity of the scenarios, the average number of instantaneously transmitted packets follows an increasing trend. The total duration of a flow that is routed from source to destination nodes gives information about the choice of bandwidth of the selected route. In this context, the duration time of some randomly selected flows from each scenario were tested. In the tested flows, particularly the long-term elephant-type flows, our proposed approach achieved the shortest duration times. Our approach optimizes one of the key factor of the QoS while also performing resource allocation. The load-balanced use of network devices during routing has a positive effect on many QoS factors. Because, the determination of non-congested routes allows the packets of the flow to be transmitted with the least networking problems. In this context, the cumulative load distribution passing over the 5 switches which are determined within the scope of the topology, has been examined for all routing methods within the Scenario 3. The proposed RL-based routing method showed a more load-balanced distribution on the switches in the long run compared to the other two methods. The load distribution of the other SDN-based routing proposal was influenced by the network element distribution connected to the switches. According to the method and results, the research questions that we asked within the scope of our study were answered as follows.

- RQ-1: Can QoS and load balancing subjects which are often handled separately, be addressed simultaneously? We offered an RL-based controller design to address QoS and load balancing research topics at the same time. The proposed method assigns three optimal weights on QoS score calculation to determine best route for requested flow. On the other hand, the QoS factor weights are selected according to the load

status of the network. System feeds each other to optimize load distribution and ensure QoS requirements of any flow. For this reason, we connected load balanced and QoS-aware approaches in a one architecture using RL domain.

- RQ-2: Is it possible to utilize the capacity of the network in a balanced way and to meet the QoS demand of the flows equally and democratically? The question is answered in a most way according to the proposed tests in Section 5.6. In the system load balance analysis, the proposed routing method achieved better load distribution for selected switches than other methods thanks to the RL. The better load distribution on the network prevents system from congestion issues and it is expected a health environment for data transfer. To test this hypothesis, some QoS factor oriented tests such as total packet loss, average jitter-delay time and bandwidth provisioning capacity are proposed and the routing methods are compared each other according to their performances. Our routing method achieved a relatively better results than other the routing methods at least two test scenarios on each performance tests. In this context, we can report that the proposed controller satisfies flow QoS needs when distributing the load in a balanced way. Another side, our routing method is performed equally or slightly poor results than equally-weighted QoS routing approach in some tests. We explain it with the two-way objective function and the long-run optimizer selection. In two-way objective function, the proposed method adaptively adjusts QoS factor weights to distribute load democratically and to meet QoS needs of each flow. Also, Q-learning method search the best results in the long-run to gather higher rewards in the next states. These two objective does not considered in other SDN based method simultaneously and the flows are routed only QoS manners. Thus, some challenges can be ignored to achieve higher performance in the large perspective for our method. To prove that, the proposed method has best scores nearly all the test in Scenario 3 that is the largest scenario than others.

The proposed routing approach is designed on the Q-learning technology which does not need excessive computational power, considers both load balancing and QoS factors while

routing according to the traffic status of the network. This work is open to development by network administrators and researchers with a reward function and action space definitions which can be designed to the needs of the network environment. It also incorporates the power of the best-effort technique into the route selection mechanism. With all these aspects, our study offers its contributions to the literature.

In the future, we plan to add an additional controller for the parallel jobs which manages system states such as delay, bandwidth. In this way, we aim for a more modular method by separating standard works from the routing. In routing decision side, it is aimed to work on a deep Q-learning method and an adaptive self-updating mechanism in order to learn and moderate network traffic more effectively. The non-linearity of the network traffic requires a comprehensive action space for QoS weight selection. The hidden layered architecture of the deep learning offers to learn sophisticated patterns in a composed way. Thus, a more advanced Q-table is obtained and can be used for better routing operations. Additionally, a self-updating mechanism is planned to implement. In this way, it is considered to offer an approach that periodically incorporates changing network traffic patterns.

REFERENCES

- [1] Cisco Annual Internet Report (2018–2023) White Paper. Technical report, Cisco Systems, Inc, **2020**.
- [2] R. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, **2018**.
- [3] Software-Defined Networking: The New Norm for Networks. Technical report, Open Networking Foundation, **2012**.
- [4] D.Kreutz, F. M. V. Ramos, and S. Azodolmolky S. Uhlig P. E. Veríssimo, C. E. Rothenberg. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, **2015**. doi:10.1109/JPROC.2014.2371999.
- [5] Component-based software defined networking framework - build sdn agilely, **2022**.
- [6] Cisco. Campus network for high availability design guide, **2008**.
- [7] J. Desjardins. How much data is generated each day?, **2019**.
- [8] N. McKeown and G. Parulkar L. Peterson J. Rexford S. Shenker J. Turner T. Anderson, H. Balakrishnan. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, **2008**. doi:10.1145/1355734.1355746.
- [9] S. Jain and J. Ong L. Poutievski A. Singh S. Venkata J. Wanderer J. Zhou M. Zhu J. Zolla U. Hölzle S. Stuart A. Vahdat A. Kumar, S. Mandal. B4: Experience with a globally-deployed software defined wan. 43(4):3–14, **2013**. ISSN 0146-4833. doi:10.1145/2534169.2486019.

- [10] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, **1996**. doi:10.1109/49.536364.
- [11] M. Karakus and A. Durresi. Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, 80:200–218, **2017**. ISSN 1084-8045. doi:https://doi.org/10.1016/j.jnca.2016.12.019.
- [12] L. Li and Q. Xu. Load balancing researches in sdn: A survey. In *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 403–408. **2017**. doi:10.1109/ICEIEC.2017.8076592.
- [13] H. Egilmez and A. M. Tekalp S. T. Dane, K. T. Bagci. Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–8. **2012**.
- [14] J. Yan and B. Liu X. Guo H. Zhang, Q. Shuai. Hiqos: An sdn-based multipath qos solution. *China Communications*, 12(5):123–133, **2015**. doi:10.1109/CC.2015.7112035.
- [15] X. Shi and T. Yang L. Zhang P. Liu H. Zhang Z. Liang Y. Li, H. Xie. An openflow-based load balancing strategy in sdn. *Computers, Materials Continua*, 61:385–398, **2019**. doi:10.32604/cmc.2020.06418.
- [16] M. C. Nkosi and S. Dlamini A. A. Lysko. Multi-path load balancing for sdn data plane. In *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, pages 1–6. **2018**. doi:10.1109/ICONIC.2018.8601241.
- [17] T. Kosugiyama and T. Hayashi K. Yamaoka K. Tanabe, H. Nakayama. A flow aggregation method based on end-to-end delay in sdn. In *2017 IEEE*

- International Conference on Communications (ICC)*, pages 1–6. **2017**. doi:10.1109/ICC.2017.7996341.
- [18] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, **1992**.
- [19] J. F. Kurose. *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, **2005**.
- [20] J. Sherry and S. Ratnasamy. A survey of enterprise middlebox deployments. **2012**.
- [21] R. Perlman. An algorithm for distributed computation of a spanningtree in an extended lan. In *Proceedings of the Ninth Symposium on Data Communications, SIGCOMM '85*, page 44–53. Association for Computing Machinery, New York, NY, USA, **1985**. ISBN 0897911644. doi:10.1145/319056.319004.
- [22] Open networking foundation. <https://opennetworking.org/>, **2017**.
- [23] A. Lazar and F. Marconcini K.S. Lim. Realizing a foundation for programmability of atm networks with the binding architecture. *Selected Areas in Communications, IEEE Journal on*, 14:1214 – 1227, **1996**. doi:10.1109/49.536363.
- [24] D.L. Tennenhouse and D.J. Wetherall G.J. Minden J.M. Smith, W.D. Sincoskie. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, **1997**. doi:10.1109/35.568214.
- [25] M. Caesar and J. Rexford A. Shaikh J. van der Merwe D. Caldwell, N. Feamster. Design and implementation of a routing control platform. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design Implementation - Volume 2, NSDI'05*, page 15–28. USENIX Association, USA, **2005**.

- [26] N. Feamster and E. Zegura J. Rexford. The road to sdn: An intellectual history of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 44(2):87–98, **2014**. ISSN 0146-4833.
- [27] O. Salman and A. Chehab I. H. Elhajj, A. Kayssi. Sdn controllers: A comparative study. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6. **2016**. doi:10.1109/MELCON.2016.7495430.
- [28] A. Binsahaq and K. Salah T. R. Sheltami. A survey on autonomic provisioning and management of qos in sdn networks. *IEEE Access*, 7:73384–73435, **2019**. doi:10.1109/ACCESS.2019.2919957.
- [29] Mininet Contributors. Mininet. <http://mininet.org/>.
- [30] J. H. Cox and J. Ivey R. J. Clark G. Riley H. L. Owen J. Chung, S. Donovan. Advancing software-defined networks: A survey. *IEEE Access*, 5:25487–25526, **2017**. doi:10.1109/ACCESS.2017.2762291.
- [31] S. Saraswat and R. Mishra A. Gupta T. Dutta V. Agarwal, H. P. Gupta. Challenges and solutions in software defined networking: A survey. *Journal of Network and Computer Applications*, 141:23–58, **2019**. ISSN 1084-8045. doi:<https://doi.org/10.1016/j.jnca.2019.04.020>.
- [32] R. Ruby J. Pan M. Tanha, D. Sajjadi. Traffic engineering enhancement by progressive migration to sdn. *IEEE Communications Letters*, 22(3):438–441, **2018**. doi:10.1109/LCOMM.2018.2789419.
- [33] N. Bizanis and F. A. Kuipers. Sdn and virtualization solutions for the internet of things: A survey. *IEEE Access*, 4:5591–5606, **2016**. doi:10.1109/ACCESS.2016.2607786.
- [34] M. Amiri and M. Abdallah H. Al Osman, S. Shirmohammadi. An sdn controller for delay and jitter reduction in cloud gaming. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, page 1043–1046. Association

for Computing Machinery, New York, NY, USA, **2015**. ISBN 9781450334594. doi:10.1145/2733373.2806397.

- [35] L Tello-Oquendo and V. Pla S. C. Lin, I. F. Akyildiz. Software-defined architecture for qos-aware iot deployments in 5g systems. *Ad Hoc Networks*, 93:101911, **2019**. ISSN 1570-8705. doi:<https://doi.org/10.1016/j.adhoc.2019.101911>.
- [36] J. M. Llopis and T. Janaszka J. Pieczerek. Minimizing latency of critical traffic through sdn. In *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–6. **2016**. doi:10.1109/NAS.2016.7549408.
- [37] M Huang and W. Xu S Guo Y. Xu W. Liang, Z. Xu. Dynamic routing for network throughput maximization in software-defined networks. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. **2016**. doi:10.1109/INFOCOM.2016.7524613.
- [38] A. Azzouni, R. Boutaba, and G. Pujolle. Neuroute: Predictive dynamic routing for software-defined networks. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–6. **2017**. doi:10.23919/CNSM.2017.8256059.
- [39] H. Xu and H. Deng H. Huang H. Wang X.Y. Li, L. Huang. Incremental deployment and throughput maximization routing for a hybrid sdn. *IEEE/ACM Transactions on Networking*, 25(3):1861–1875, **2017**. doi:10.1109/TNET.2017.2657643.
- [40] M. Amiri and S. Shirmohammadi H. Al Osman. Game-aware and sdn-assisted bandwidth allocation for data center networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 86–91. **2018**. doi:10.1109/MIPR.2018.00023.

- [41] C. C. Lin and W. B. Chen H. H. Chin. Balancing latency and cost in software-defined vehicular networks using genetic algorithm. *Journal of Network and Computer Applications*, 116, **2018**. doi:10.1016/j.jnca.2018.05.002.
- [42] C. Lin and Z. Liu S. Jia J. Zhu Y. Bi, H. Zhao. Dte-sdn: A dynamic traffic engineering engine for delay-sensitive transfer. *IEEE Internet of Things Journal*, 5(6):5240–5253, **2018**. doi:10.1109/JIOT.2018.2872439.
- [43] T. Radusinovic S. Tomovic. Fast and efficient bandwidth-delay constrained routing algorithm for sdn networks. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 303–311. **2016**. doi:10.1109/NETSOFT.2016.7502426.
- [44] C. Lin and G. Deng K. Wang. A qos-aware routing in sdn hybrid networks. *Procedia Computer Science*, 110:242–249, **2017**. ISSN 1877-0509. doi:https://doi.org/10.1016/j.procs.2017.06.091. 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops.
- [45] S. Xu and J. Ren S. Wang X. Wang, G. Yang. Routing optimization for cloud services in sdn-based internet of things with team capacity constraint. *Journal of Communications and Networks*, 22(2):145–158, **2020**. doi:10.1109/JCN.2020.000006.
- [46] R. Boutaba and S. Ayoubi N. Shahriar F. Estrada-Solano O. Caicedo Rendon M. Salahuddin, N. Limam. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9, **2018**. doi:10.1186/s13174-018-0087-2.
- [47] H. Ghalwash and C. H. Huang. A qos framework for sdn-based networks. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 98–105. **2018**. doi:10.1109/CIC.2018.00024.

- [48] T. Chin and K. Xiong M. Rahouti. Applying software-defined networking to minimize the end-to-end delay of network services. *SIGAPP Appl. Comput. Rev.*, 18(1):30–40, **2018**. ISSN 1559-6915. doi:10.1145/3212069.3212072.
- [49] S. Kumar and S. Virendra G. Bansal, S. Gaurang. A machine learning approach for traffic flow provisioning in software defined networks. In *2020 International Conference on Information Networking (ICOIN)*, pages 602–607. **2020**. doi:10.1109/ICOIN48656.2020.9016529.
- [50] J. Chavula and H. Suleman M Densmore. Using sdn and reinforcement learning for traffic engineering in ubuntunet alliance. In *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 349–355. **2016**. doi:10.1109/ICACCE.2016.8073774.
- [51] I. A. Akyildiz P. Wang M. Luo S. C. Lin, and. Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 25–33. **2016**. doi:10.1109/SCC.2016.12.
- [52] S. Sendra and J. M. Jimenez O. Romero A. Rego, J. Lloret. Including artificial intelligence in a routing protocol using software defined networks. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 670–674. **2017**. doi:10.1109/ICCW.2017.7962735.
- [53] G. Stampa and V. Muntés-Mulero A. Cabellos M. Arias, D. Sánchez-Charles. A deep-reinforcement learning approach for software-defined networking routing optimization. *arXiv preprint arXiv:1709.07080*, **2017**.
- [54] F. Francois and E. Gelenbe. Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 283–288. **2016**. doi:10.1109/MASCOTS.2016.26.

- [55] D. Casas-Velasco and N. Fonseca O. C. Rendon. Intelligent routing based on reinforcement learning for software-defined networking. *IEEE Transactions on Network and Service Management*, 18(1):870–881, **2021**. doi:10.1109/TNSM.2020.3036911.
- [56] A. Al-Jawad and O. Gemikonakli R. Trestian I. S. Comşa, P. Shah. Redo: A reinforcement learning-based dynamic routing algorithm selection method for sdn. In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 54–59. **2021**. doi:10.1109/NFV-SDN53031.2021.9665140.
- [57] A. Montazerolghaem and M. H. Yaghmaee. Load-balanced and qos-aware software-defined internet of things. *IEEE Internet of Things Journal*, 7(4):3323–3337, **2020**. doi:10.1109/JIOT.2020.2967081.
- [58] X. Guo and M. Peng H. Lin, Z. Li. Deep reinforcement learning based qos-aware secure routing for sdn-iot. *IEEE Internet of Things Journal*, PP:1–1, **2019**. doi:10.1109/JIOT.2019.2960033.
- [59] M. Hosseinzadeh A. Rezaee A. A. Neghabi, N. Jafari. Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature. *IEEE Access*, 6:14159–14178, **2018**. doi:10.1109/ACCESS.2018.2805842.
- [60] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier, New York, **1976**.
- [61] T. H. Cormen and C. Stein C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms, 3rd Edition*. MIT Press, **2009**. ISBN 9780262533058.
- [62] L Liao and V. C. M. Leung. Lldp based link latency monitoring in software defined networks. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 330–335. **2016**. doi:10.1109/CNSM.2016.7818442.

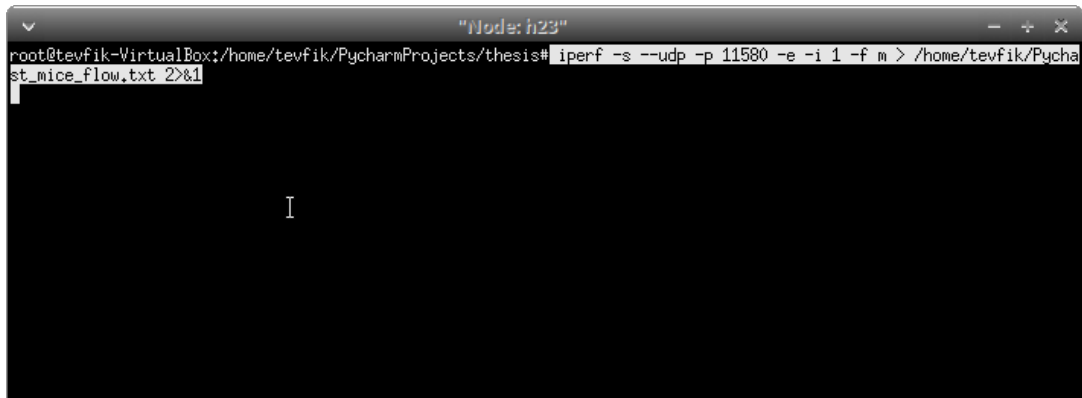
- [63] Z. Shu and S. Wang D. Li S. Rho C. Yang J. Wan, J. Lin. Traffic engineering in software-defined networking: Measurement and management. *IEEE Access*, 4:3246–3256, **2016**. doi:10.1109/ACCESS.2016.2582748.
- [64] C. Hedrick. Routing Information Protocol. RFC 1058, **1988**. doi:10.17487/RFC1058.
- [65] L. Al Shalabi and Z. Shaaban. Normalization as a preprocessing engine for data mining and the approach of preference matrix. In *2006 International Conference on Dependability of Computer Systems*, pages 207–214. **2006**. doi:10.1109/DEPCOS-RELCOMEX.2006.38.
- [66] R. L. S. de Oliveira and L. R. Prete C. M. Schweitzer, A.A Shinoda. Using mininet for emulation and prototyping software-defined networks. In *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6. **2014**. doi:10.1109/ColComCon.2014.6860404.
- [67] V. Guéant. iperf - the tcp, udp and sctp network bandwidth measurement tool, **2014**.
- [68] K. Papagiannaki and P. Thiran K. Salamatian C. Diot N. Taft, S. Bhattacharyya. A pragmatic definition of elephants in internet backbone traffic. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, page 175–176. Association for Computing Machinery, New York, NY, USA, **2002**. doi:10.1145/637201.637227.
- [69] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, **2003**. doi:10.1145/859716.859719.
- [70] K. C. Lan and J. Heidemann. A measurement study of correlation of internet flow characteristics. *Computer Networks*, 17:46–62, **2006**. doi:10.1016/j.comnet.2005.02.008.

- [71] Y. Zhang and S. Shenker L. Breslau, V. Paxson. On the characteristics and origins of internet flow rates. *SIGCOMM Comput. Commun. Rev.*, 32(4):309–322, **2002**. doi:10.1145/964725.633055.
- [72] J. Bergstra and B. Kégl R. Bardenet, Y. Bengio. Algorithms for hyper-parameter optimization. In P. Bartlett F. Pereira K.Q. Weinberger J. Shawe-Taylor, R. Zemel, editor, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., **2011**.

Appendix B

iperf Flow Report

Figure 6.5 shows the data transfer report created with the *iperf* network traffic generation tool. In a virtual topology which is created using Mininet, network traffic is generated between the source and target hosts using *iperf*. Host 23, which is determined as the target host in Figure 6.3, receives the data that is sent from host 13, which is defined as the source host in Figure 6.4. On the target host, listened port is specified for the communication. On the other hand, the size of the data to be sent and the amount of bandwidth are specified on the source host. In addition, the logging period of the receiving data is determined with `-i` host 23. The transferred data from source host to target host is periodically recorded and generates the report in Figure 6.5. The data size, bandwidth, jitter, average delay time, total number of packets, number of lost packets and network power values are obtained in a second period. These reports are generated for each flow in the test scenarios. At the end, They are used to measure performance of tested approaches.

A terminal window titled "Node: h23" showing a command prompt. The prompt is "root@tevfik-VirtualBox:/home/tevfik/PycharmProjects/thesis#". The command entered is "iperf -s --udp -p 11580 -e -i 1 -f m > /home/tevfik/PycharmProjects/thesis/mice_flow.txt 2>&1". The terminal output is currently blank, with a cursor visible on the line below the command.

```
root@tevfik-VirtualBox:/home/tevfik/PycharmProjects/thesis# iperf -s --udp -p 11580 -e -i 1 -f m > /home/tevfik/PycharmProjects/thesis/mice_flow.txt 2>&1
```

Figure 6.3 Flow initialization command on receiver host using *iperf* tool

```
root@tevfik-VirtualBox:/home/tevfik/PycharmProjects/thesis# iperf -c 10.0.0.23 --udp -e -p 11580 -b
-----
Client connecting to 10.0.0.23, UDP port 11580 with pid 15018
Binding to local address 10.0.0.13
Sending 1470 byte datagrams, IPG target: 3738.40 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 5] local 10.0.0.13 port 8328 connected with 10.0.0.23 port 11580
[ ID] Interval      Transfer    Bandwidth   Write/Err   PPS
[ 5] 0.0000-26.3183 sec 9.00 MBytes 2.87 Mbits/sec 6420/0      243 pps
[ 5] Sent 6420 datagrams
[ 5] Server Report:
[ 5] 0.0000-26.3177 sec 9.00 MBytes 2.87 Mbits/sec 0.258 ms 0/ 6420 (0%) 1.522/ 0.234/71.14
[ 5] 0.0000-26.3177 sec 3 datagrams received out-of-order
root@tevfik-VirtualBox:/home/tevfik/PycharmProjects/thesis#
```

Figure 6.4 Flow initialization command on sender host using *iperf* tool

```

Server listening on UDP port 11580 with pid 15015
Receiving 1470 byte datagrams
UDP buffer size: 0.20 MByte (default)
-----
[ 5] local 10.0.0.23 port 11580 connected with 10.0.0.13 port 8328
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total   Latency avg/min/max/ stdev  pps  NetPwr
[ 5] 0.0000-1.0000 sec 0.34 MBytes  2.88 Mb/s/sec  1.761 ms    0/245 (0%)  2.338/ 0.283/17.731/ 2.535 ms  244 pps  154.06
[ 5] 1.0000-2.0000 sec 0.33 MBytes  2.76 Mb/s/sec  2.493 ms    0/235 (0%)  2.149/ 0.281/15.790/ 2.506 ms  236 pps  160.71
[ 5] 1.0000-2.0000 sec 1 datagrams  received out-of-order
[ 5] 2.0000-3.0000 sec 0.33 MBytes  2.74 Mb/s/sec  1.920 ms    0/233 (0%)  2.602/ 0.289/21.877/ 2.711 ms  233 pps  131.61
[ 5] 3.0000-4.0000 sec 0.35 MBytes  2.92 Mb/s/sec  1.667 ms    0/253 (0%)  1.431/ 0.274/13.502/ 1.998 ms  253 pps  259.89
[ 5] 4.0000-5.0000 sec 0.23 MBytes  1.92 Mb/s/sec  4.838 ms    0/163 (0%)  1.886/ 0.275/50.689/ 5.377 ms  162 pps  127.05
[ 5] 5.0000-6.0000 sec 0.34 MBytes  2.81 Mb/s/sec  2.987 ms    0/239 (0%)  1.535/ 0.258/24.966/ 2.751 ms  241 pps  228.91
[ 5] 6.0000-7.0000 sec 0.29 MBytes  2.41 Mb/s/sec  3.744 ms    0/205 (0%)  3.291/ 0.263/71.145/ 5.800 ms  204 pps  91.57
[ 5] 7.0000-8.0000 sec 0.32 MBytes  2.68 Mb/s/sec  3.020 ms    0/228 (0%)  2.507/ 0.283/21.629/ 2.803 ms  229 pps  133.67
[ 5] 8.0000-9.0000 sec 0.36 MBytes  3.01 Mb/s/sec  0.376 ms    0/256 (0%)  1.290/ 0.247/15.654/ 1.964 ms  257 pps  291.76
[ 5] 9.0000-10.0000 sec 0.35 MBytes  2.98 Mb/s/sec  1.130 ms    0/253 (0%)  0.954/ 0.246/17.190/ 1.889 ms  253 pps  389.92
[ 5] 10.0000-11.0000 sec 0.32 MBytes  2.69 Mb/s/sec  4.194 ms    0/229 (0%)  2.511/ 0.292/18.235/ 2.776 ms  227 pps  134.07
[ 5] 11.0000-12.0000 sec 1 datagrams  received out-of-order
[ 5] 12.0000-13.0000 sec 0.36 MBytes  3.02 Mb/s/sec  2.029 ms    0/257 (0%)  0.764/ 0.262/18.235/ 1.445 ms  258 pps  494.73
[ 5] 13.0000-14.0000 sec 0.34 MBytes  2.87 Mb/s/sec  1.862 ms    0/244 (0%)  1.531/ 0.275/14.275/ 2.094 ms  242 pps  234.27
[ 5] 14.0000-15.0000 sec 0.36 MBytes  3.05 Mb/s/sec  0.056 ms    0/259 (0%)  1.115/ 0.243/10.209/ 1.610 ms  262 pps  341.55
[ 5] 15.0000-16.0000 sec 0.37 MBytes  3.09 Mb/s/sec  2.181 ms    0/263 (0%)  0.829/ 0.256/15.615/ 1.476 ms  263 pps  466.24
[ 5] 16.0000-17.0000 sec 0.32 MBytes  2.65 Mb/s/sec  3.292 ms    0/225 (0%)  2.515/ 0.261/23.017/ 3.216 ms  222 pps  131.50
[ 5] 17.0000-18.0000 sec 0.35 MBytes  2.94 Mb/s/sec  0.169 ms    0/250 (0%)  1.243/ 0.255/14.988/ 1.711 ms  253 pps  295.73
[ 5] 18.0000-19.0000 sec 0.35 MBytes  2.92 Mb/s/sec  1.161 ms    0/248 (0%)  1.434/ 0.272/33.488/ 2.743 ms  248 pps  254.22
[ 5] 19.0000-20.0000 sec 0.37 MBytes  3.14 Mb/s/sec  0.080 ms    0/267 (0%)  0.990/ 0.255/ 6.748/ 1.236 ms  267 pps  396.28
[ 5] 20.0000-21.0000 sec 1 datagrams  received out-of-order
[ 5] 21.0000-22.0000 sec 0.37 MBytes  3.08 Mb/s/sec  0.667 ms    0/262 (0%)  0.921/ 0.234/12.492/ 1.342 ms  262 pps  417.95
[ 5] 22.0000-23.0000 sec 0.36 MBytes  3.01 Mb/s/sec  3.320 ms    0/256 (0%)  1.257/ 0.236/12.899/ 1.655 ms  256 pps  299.47
[ 5] 23.0000-24.0000 sec 0.33 MBytes  2.79 Mb/s/sec  0.263 ms    0/237 (0%)  0.759/ 0.273/13.152/ 1.306 ms  262 pps  507.59
[ 5] 24.0000-25.0000 sec 0.33 MBytes  2.79 Mb/s/sec  0.548 ms    0/232 (0%)  1.936/ 0.267/17.810/ 2.539 ms  236 pps  179.94
[ 5] 25.0000-26.0000 sec 0.37 MBytes  3.09 Mb/s/sec  0.084 ms    0/263 (0%)  0.823/ 0.254/12.552/ 1.076 ms  264 pps  469.54
[ 5] 26.0000-27.0000 sec 0.37 MBytes  3.09 Mb/s/sec  0.066 ms    0/263 (0%)  0.892/ 0.261/ 9.431/ 1.136 ms  262 pps  433.41
[ 5] 27.0000-28.0000 sec 0.35 MBytes  2.92 Mb/s/sec  4.956 ms    0/248 (0%)  1.711/ 0.257/23.342/ 2.384 ms  243 pps  213.11
[ 5] 28.0000-29.0000 sec 9.00 MBytes  2.87 Mb/s/sec  0.259 ms    0/6420 (0%)  1.522/ 0.234/71.145/ 2.524 ms  243 pps  235.65
[ 5] 29.0000-30.0000 sec 3 datagrams  received out-of-order

```

Figure 6.5 Periodic data transfer report that is written in receiver host using *iperf* tool

Appendix C

Additional Tuning Evaluation

The CMA calculation of the reward value is also examined for the appropriate RL model selection. Below, each test figure shows the performance results of the models at different lr values, while keeping the γ value constant. When we evaluate the results, it is seen that when γ and lr are determined as 0.1, the highest CMA of reward is obtained in the long run.

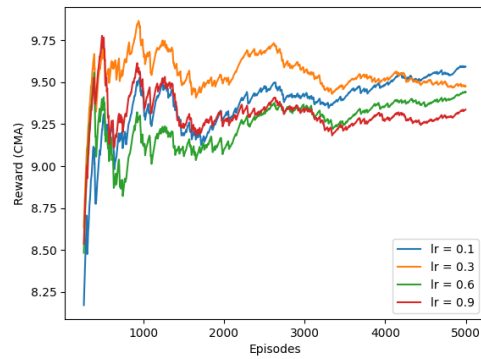
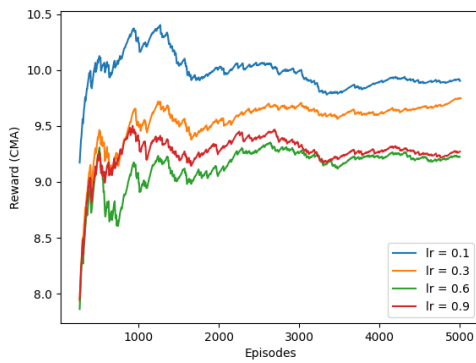


Figure 6.6 CMA of R for different lr , $\gamma = 0.1$ Figure 6.7 CMA of R for different lr , $\gamma = 0.3$

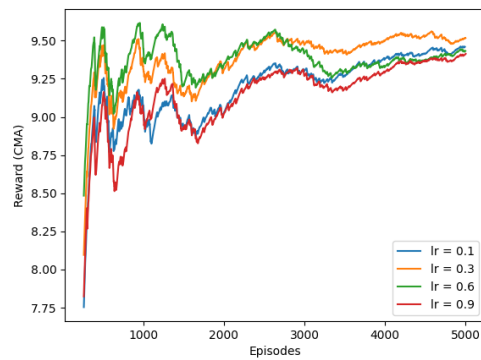
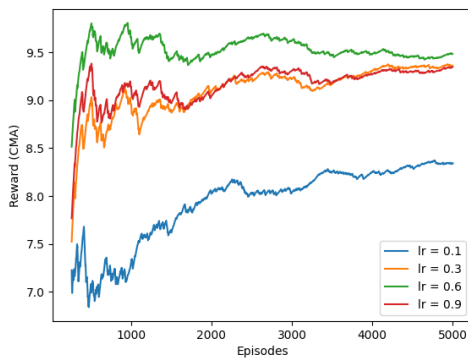


Figure 6.8 CMA of R for different lr , $\gamma = 0.6$ Figure 6.9 CMA of R for different lr , $\gamma = 0.9$