A DAG (DIRECTED ACYCLIC GRAPH)-BASED DLT (DISTRIBUTED LEDGER TECHNOLOGY) INTEGRATED CROSS-NETWORK QOS-MOTIVATED TRAFFIC MANAGEMENT FRAMEWORK USING REINFORCEMENT LEARNING (RL) IN SOFTWARE DEFINED NETWORKS (SDNS)

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ANKARA UNIVERSITY

by

Barış KURTULUŞ

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN
ARTIFICIAL INTELLIGENCE TECHNOLOGIES

ANKARA
2025

ABSTRACT

Master Thesis

A DAG (DIRECTED ACYCLIC GRAPH)-BASED DLT (DISTRIBUTED LEDGER TECHNOLOGY) INTEGRATED CROSS-NETWORK QOS-MOTIVATED TRAFFIC MANAGEMENT FRAMEWORK USING REINFORCEMENT LEARNING (RL) IN SOFTWARE DEFINED NETWORKS (SDNS)

Barış KURTULUŞ

Ankara University
Graduate School of Natural and Applied Sciences
Artificial Intelligence Technology

Supervisor: Associate Prof. Dr. Murat KARAKUŞ

Modern networks require adaptive traffic management to handle increasing data volume and diverse Quality of Service (QoS) requirements. Software-Defined Networks (SDNs) offer flexibility; however, traditional routing algorithms such as Open Shortest Path First (OSPF) and Dijkstra's algorithm lack real-time adaptability, limiting QoS optimization. To address these limitations, this study introduces the Directed Acyclic Graph (DAG)-based QoS-Centric traffic management framework (DQRL), integrating DAG-based Distributed Ledger Technology (DLT) and Reinforcement Learning (RL) for efficient QoS-aware routing.

DQRL dynamically selects optimal paths based on bandwidth, delay, and packet loss, ensuring high-throughput and low-latency routing. Unlike blockchain-based methods, DAG enables blockless, decentralized transactions, facilitating rapid validation and scalability. RL, particularly Q-learning, allows adaptive path selection by learning from network conditions.

Simulations were conducted using Mininet, Python, and MATLAB in networks with 5 to 10 Autonomous Systems (ASes), each containing 5 to 10 nodes. The framework was evaluated against Decentralized Routing Method (DRM) and Pyramidal Routing Method (PRM) using five Key Performance Indicators (KPIs): Flow Establishment Time (FET), Messages Exchanged and Executed (MEE), Average Pathlet Length (APL), Average Node Length (ANoL), and Average Network Length (ANL). Results demonstrate that DQRL reduces FET by 29%, achieves up to 42% improvement over PRM, and lowers MEE by 30%, reducing message overhead by 23% compared to Blockchain-based QoS Routing Method (BCQRM). Additionally, DQRL achieves shorter paths with reduced APL, ANoL, and ANL, ensuring efficient routing.

By integrating DAG-based DLT and RL, DQRL surpasses traditional and blockchain-enhanced routing methods, offering a scalable, high-speed solution for SDNs.

February 2025, 135 pages
Key Words: Directed Acyclic Graph, Distributed Ledger Technology, Routing, Quality of Service, Reinforcement Learning, Software Defined Networks, Traffic Management

ÖZET

Yüksek Lisans Tezi

YAZILIM TANIMLI AĞLAR (SDN) İÇİNDE TAKVİYELİ ÖĞRENME (RL) KULLANARAK YÖNLENDİRİLMİŞ DÖNGÜSÜZ GRAFİK (DAG) TEMELLİ DAĞITIK DEFTER TEKNOLOJİSİ (DLT) ENTEGRELİ HİZMET KALİTESİ (QOS) ODAKLI AĞLAR ARASI TRAFİK YÖNETİM ÇERÇEVESİ

Barış KURTULUŞ

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Yapay Zeka Teknolojileri Anabilim Dalı

Danışman: Doç. Dr. Murat KARAKUŞ

Modern ağlarda artan veri hacmi ve farklı Hizmet Kalitesi (QoS) gereksinimleri, adaptif trafik yönetimini zorunlu hale getirmektedir. Yazılım Tanımlı Ağlar (SDN) esneklik sağlasa da, Açık En Kısa Yol Öncelikli Yönlendirme (OSPF) ve Dijkstra algoritması gibi geleneksel yönlendirme yöntemleri, gerçek zamanlı uyarlanabilirlikten yoksundur ve QoS optimizasyonunu sınırlamaktadır. Bu çalışma, Yönlendirilmiş Asiklik Grafik (DAG)-tabanlı QoS Odaklı Trafik Yönetim Çerçevesi (DQRL) adlı yeni bir model önermektedir. DQRL, DAG-tabanlı Dağıtık Defter Teknolojisi (DLT) ve Pekiştirmeli Öğrenme (RL) kullanarak merkeziyetsiz ve QoS uyumlu yönlendirme sağlamayı hedeflemektedir.

DQRL, bant genişliği, gecikme ve paket kaybı gibi metriklere dayalı olarak en uygun yolları dinamik şekilde seçerek yüksek verimli ve düşük gecikmeli yönlendirme sunar. Blok zinciri tabanlı yöntemlerden farklı olarak DAG, blok içermeyen merkeziyetsiz işlemleri destekleyerek hızlı doğrulama ve ölçeklenebilirlik sağlar. RL, özellikle Q-öğrenme (Q-learning) ile ağ koşullarını analiz ederek adaptif yol seçimi yapmaktadır.

Simülasyonlar, 5 ila 10 Otonom Sistem (AS) içeren ağlarda gerçekleştirilmiş olup, her AS 5 ila 10 düğüm içermektedir. Mininet, Python ve MATLAB kullanılarak yapılan deneylerde, DQRL, Merkeziyetsiz Yönlendirme Yöntemi (DRM) ve Piramidal Yönlendirme Yöntemi (PRM) ile karşılaştırılmıştır. Akış Kurulum Süresi (FET), Mesaj Değişimi ve İşleme (MEE) ve Ortalama Yol Uzunluğu (APL) metrikleri üzerinden yapılan analizlerde DQRL'nin FET'i %29 azalttığı, PRM'e kıyasla %42'ye varan iyileşme sağladığı, MEE'yi %30 ve mesaj yükünü BCQRM'ye kıyasla %23 düşürdüğü görülmüştür.

DAG-tabanlı DLT ve RL entegrasyonu sayesinde, DQRL, geleneksel ve blok zinciri destekli yönlendirme yöntemlerinden daha üstün performans sunarak SDN'ler için ölçeklenebilir, yüksek hızlı ve adaptif bir çözüm sağlamaktadır**.**

Şubat 2025, 135 sayfa
Anahtar Kelimeler: Yönlendirilmiş Asiklik Grafik, Dağıtık Defter Teknolojisi, Yönlendirme, Hizmet Kalitesi, Pekiştirmeli Öğrenme, Yazılım Tanımlı Ağlar, Trafik Yönetimi

FOREWORD AND ACKNOWLEDGEMENTS

I would like to extend my endless thanks to my advisor Associate Prof. Dr. Murat KARAKUŞ for his valuable guidance and support throughout the research process, whose knowledge and experience I benefited from during the realization of this thesis. He supported me with his opinions at every stage of the research, always made me feel his sincerity, and guided me in the right direction.

I am also deeply grateful to Prof. Dr. Asım Egemen YILMAZ, the head of the department, for his support and encouragement throughout my academic journey.

Additionally, I wish to express my gratitude to the faculty members of Ankara University Graduate School of Natural and Applied Sciences for their support and help.

I would also like to express my sincere gratitude to the Scientific and Technological Research Council of Türkiye (TÜBİTAK) for its support under Grant No. 120E448, which has significantly contributed to the completion of this thesis.

Lastly, I would like to thank my family for their unwavering support throughout my study and for never losing their trust in me.

Barış KURTULUŞ
Ankara, February 2025

TABLE OF CONTENTS

SYMBOLS AND ABBREVIATIONS

ACLs	Access Control Lists
ANL	Average Network Length
ANoL	Average Node Length
APL	Average Pathlet Length
AS	Autonomous System
B_k	A collection of boundary nodes within the k-th Autonomous System (AS)
B5G	Beyond 5G
BCQRM	Blockchain-Based QoS Routing Method
BRo	Main controller named 'Broker'
BRo	Main controller named 'Broker'
BSDEI	Blockchain-Assisted Software-Defined Energy Internet
C_g	The collection of ASes involved in the E2E path computation for a particular SD_g.
CAPEX	Capital Expenses
C-DAG	Community-Assisted DAG
CNNs	Convolutional Neural Networks
DAG	Directed Acyclic Graph
DB-SCS	DRL and Blockchain-Based Spatial Crowdsourcing System
DDoS	Distributed Denial of Service
DDoS	Distributed Denial of Service
DiffServ	Differentiated Services
DLT	Distributed Ledger Technology
DoS	Denial-of-Service
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DRM	Decentralized Routing Method
DROM	Deep Deterministic Policy Gradient Routing Optimization Mechanism
EI	The Energy Internet
F	OpenFlow 'flow_mod' message
FE	Forward Efficiency
FET	Flow Establishment Time

I_c	Initiating controller for flow f
I_h	Initiating host for flow f
I_sw	Initiating switch for flow f
IDS	Intrusion Detection Systems
IN-QRP	Inter-Network QoS-Aware Routing Problem
IntServ	Integrated Services
IoT	Internet of Things
IoV	Internet of Vehicles
KDN	Knowledge-Defined Networking
KPIs	Key Performance Indicators
M	Local path request message
MA2C	Multi-Agent Advantage Actor-Critic
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
MEE	Messages Exchanged and Executed
N	The collection of ASes configured in the simulation environment.
N_g	The total of ASes that constitute the E2E path for a specific Service Demand (SD_g).
Next_c	Next Controller for flow f
NGNs	Next-Generation Networks
OPEX	Operational Expenses
OSPF	Open Shortest Path First
P_dem	Pathlet Demand
P_f	First packet of flow f
P_in	OpenFlow 'packet_inbound' message
P_rep	Pathlet Reply
PoS	Proof of Stake
Prev_c	Previous controller for flow f
PRM	Pyramidal Routing Method
QoS	Quality of Service
RL	Reinforcement Learning
RLMR	Reinforcement Learning-Based Multipath Routing
S_dem	Service Demand
S_rep	Service Reply
SC2	Smart Contract Chain

SDIIoT	Software-Defined IIoT
SDNs	Software-Defined Networks
SPF	Shortest Path First
Swi	The aggregate count of switches configured within the simulation parameters.
Swk_g	The number of switches present along the E2E path for SD_g within the k-th AS.
T(a)broad_b->c	Broadcast time for message (a) from (b) to (c).
T(a)exe_b	Execution time for a message (a) within (b).
T(path_E2E)deter_BR	Detemination time for an E2E path for the Service Demand in the Broker.
T(path_E2E_dlt)deter_b	Detemination time for an E2E path for the SD using DLT at (b)
T(path_L)deter_b	Detemination time for a local path from a switch to nodes at (b)
T_c	Target controller for flow f
TE	Traffic Engineering
TSC	Traffic Signal Control

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Over the past few years, global data traffic has grown considerably, primarily due to the increasing adoption of internet-capable devices, live applications, and cloud-based technologies. Cisco's annual internet report highlights that global IP traffic was anticipated to hit 396 exabytes monthly by 2022, representing a remarkable rise compared to prior years. Notably, video content was forecasted to account for 82% of this traffic (Cisco, 2022). The rapid increase in global IP traffic has also been mirrored by the growing adoption of Distributed Ledger Technology (DLT). According to a report by (Statista, 2024), the DLT market is projected to reach $103 billion by 2030, driven by its applications in secure and efficient data management. Figure 1.1 illustrates the growth trajectory of the DLT market, showcasing its potential impact across industries, including network management.

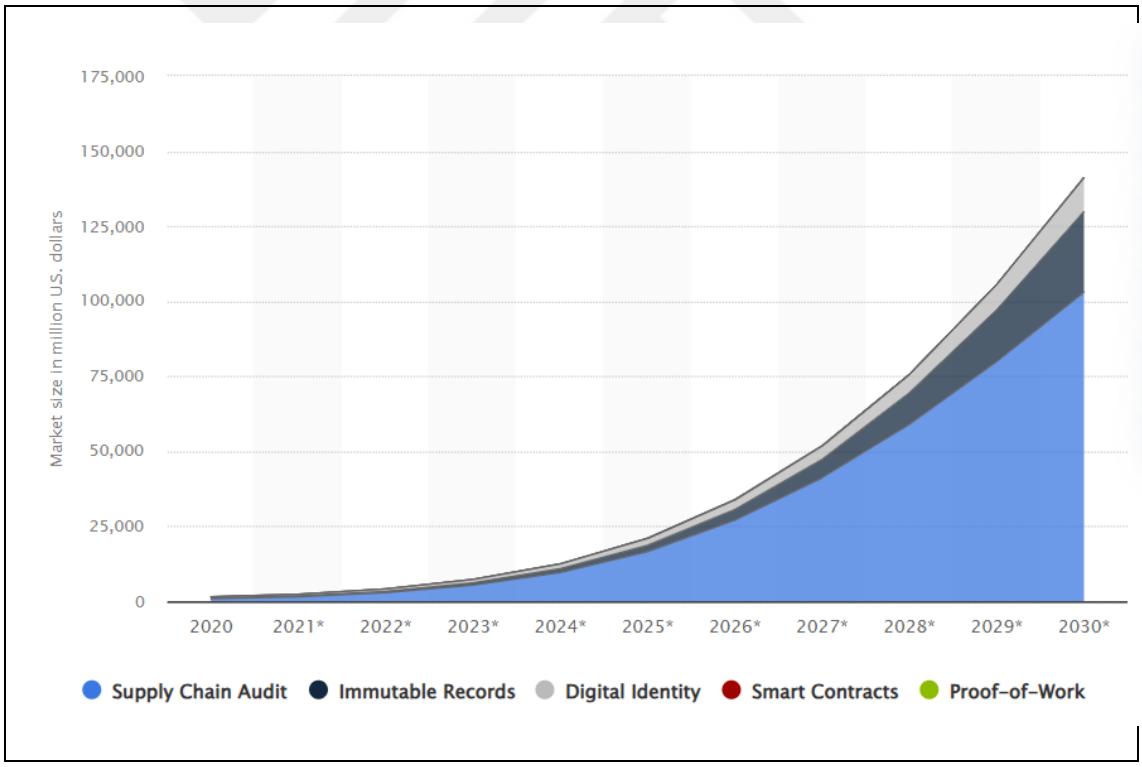


Figure 1.1 DLT market growth forecast (2020-2030) by Statista

The Figure 1.2 illustrates the global Software Defined Network (SDN) market growth forecast (MarketsandMarkets, 2023), highlighting a Compound Annual Growth Rate (CAGR) of 19.7% from 2023 to 2028. The market is projected to expand from $24.5 billion in 2023 to $60.2 billion by 2028, driven by increasing adoption in data centers, enterprise networks, and wide-area networks. Regional analysis shows North America leading in SDN adoption, followed by significant growth potential in the Asia-Pacific region. This trend underscores the growing demand for scalable, programmable, and efficient network solutions that SDN technologies offer.

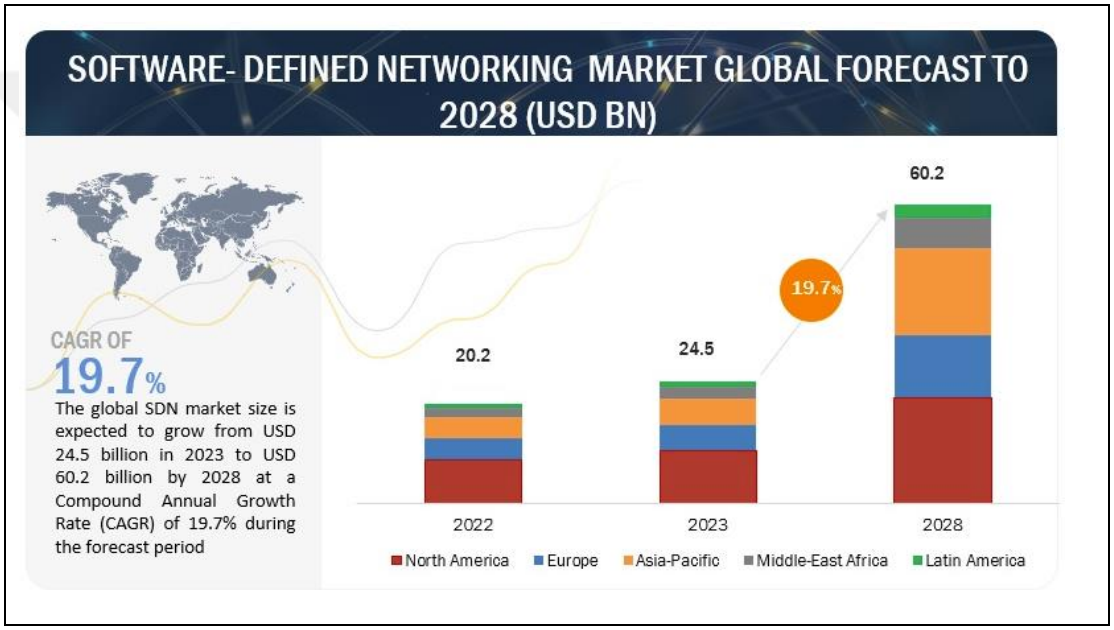


Figure 1.2 Global SDN market growth forecast (2023-2028)

The rapid increase in data traffic has placed significant pressure on network infrastructures, particularly as applications with high data demands necessitate strict QoS criteria, such as optimal bandwidth, minimal latency, and reduced packet loss. Traditional network architectures, however, rely heavily on static configurations and fixed routing algorithms, like Dijkstra's and Open Shortest Path First (OSPF), which inherently lack adaptability to meet dynamic and large-scale Quality of Service (QoS) demands (Meneguette, 2020). Traditional network architectures are fundamentally constrained by their inability to dynamically adapt to real-time traffic conditions. These limitations manifest in the form of static routing protocols, inefficient resource allocation, and poor

scalability. The primary challenges faced by traditional networks include high latency, packet loss, and limited flexibility in managing dynamic traffic patterns.

As an answer to these challenges, SDNs have emerged as a transformative approach in network management, allowing centralized control, programmability, and on-demand configuration of network resources. By decoupling the control and data Layers, SDNs provide more agile and flexible traffic management options (Agarwal et al., 2013). SDNs address these limitations by centralizing network intelligence, offering programmable control Layers, and decoupling the control and data Layers. These features enable improved traffic engineering, real-time policy enforcement, and enhanced QoS management. The SDN market itself has seen substantial growth, with a projected market size of approximately $25 billion by 2025, primarily driven by the technology's applications in data centers, enterprise environments, and wide-area networks (MarketsandMarkets, 2021).

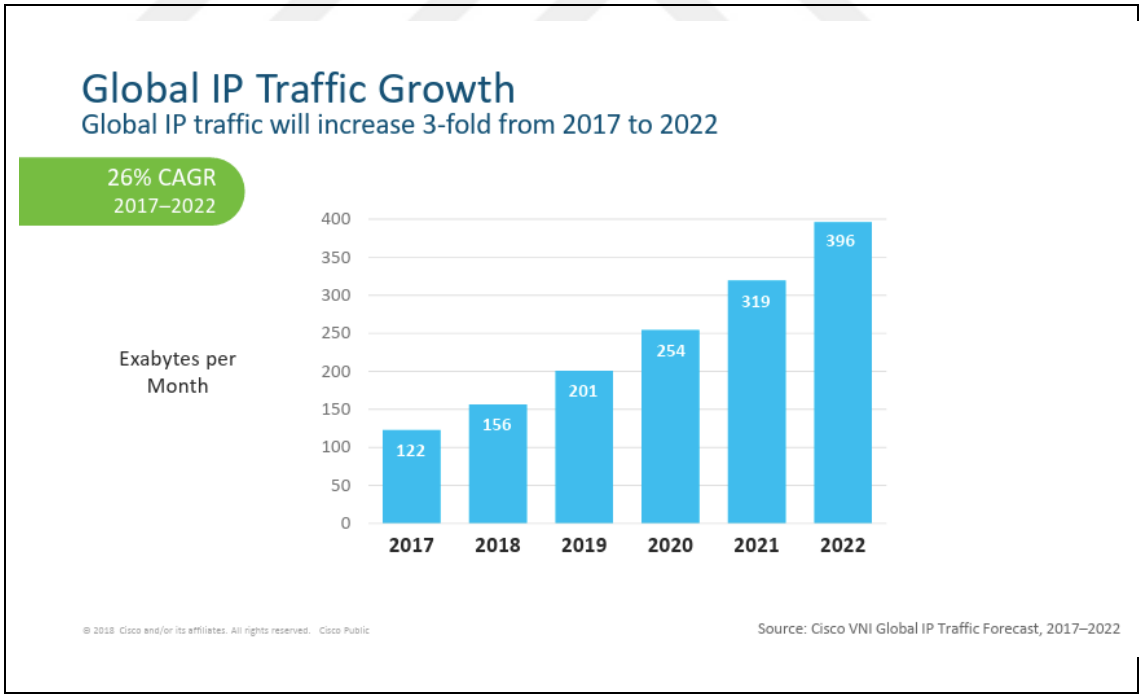


Figure 1.3 Global IP traffic growth by device type (2017-2022)

Figure 1.3 derived from Cisco VNI Global IP Traffic Forecast, presents the rapid growth in global IP traffic, which reached 396 exabytes per month by 2022, with an annual

CAGR of 26% from 2017 to 2022. Smartphones accounted for the largest traffic share (44%), followed by PCs (19%) and TVs (24%). The exponential growth is primarily driven by video content, constituting 82% of total IP traffic. This trend reflects the critical need for robust network infrastructures that can accommodate high-bandwidth, low-latency, and QoS-driven applications in modern networks. However, despite their advantages, traditional SDN routing protocols are still limited in real-time adaptability, making it challenging to meet QoS requirements across dynamic network scenarios.

In addition to SDN, DLThas garnered attention for its potential in decentralized data validation and security. Traditional blockchain, however, with its linear and sequential structure, has limitations in scaling efficiently for high-throughput environments due to its bottleneck-prone validation process (Popov, 2017). Directed Acylic Graph (DAG)-based DLT presents a promising alternative, offering a non-linear, acyclic structure that enables faster and more scalable transaction validation. Unlike conventional blockchain, DAG technology supports parallel validations, significantly enhancing scalability and making it suitable for high-speed applications like SDN, where real-time performance is critical (Popov, 2017). DLT's ability to decentralize transaction validation and enhance security makes it a powerful tool for modern networking. By employing DAG-based structures, DLT supports parallel validation, which is critical for reducing latency and ensuring real-time adaptability in SDNs. Table 2 outlines the comparative advantages of DAG-based DLT over traditional blockchain systems, emphasizing its applicability in high-throughput environments.

To address the limitations of both traditional SDN routing and blockchain-based DLT, this thesis introduces a novel traffic management strategy that integrates DAG-based DLT with Reinforcement Learning (RL). RL, particularly through algorithms like Q-learning, enables adaptive and intelligent routing based on real-time network feedback (Chen et al., 2020). RL-based approaches dynamically adjust to network changes, optimizing routing paths in response to current network conditions. By combining DAG structures with RL, the proposed approach offers a decentralized and adaptable solution for managing traffic, catering to the evolving needs of modern network environments. This thesis aims to develop a flexible and QoS-oriented traffic management framework

that leverages DAG-based DLT for decentralized validation alongside RL for dynamic and efficient pathfinding within SDNs. The evaluation phase benchmarks the framework against established methods, employing KPIs such as Flow Establishment Time (FET), Messages Exchanged and Executed (MEE), Average Pathlet Length (APL), Average Node Length (ANoL), Average Network Length (ANL) and total network throughput to validate its efficacy and scalability.

1.1 Motivation

In recent years, modern communication networks have confronted with severe issues owing to explosive growth of data traffic based on the volume of internet-capable devices, real-time applications and cloud services. The static routing algorithms (like Dijkstra's or OSPF), which are the basis of traditional network infrastructures, turn out to be inadequate and unsuited for coping with increasing needs of QoS in modern large-scale and complex environments. Such legacy routing protocols are already seen struggling to optimize essential QoS parameters (e.g., bandwidth, latency, packet loss), especially in more dynamic and high-traffic scenarios such as SDN and Network Functions Virtualization (NFV).

SDNs present a groundbreaking approach to overcoming these challenges by introducing centralized oversight and configurable network switches. The decoupling of the control and data Layers in SDNs enables precise and efficient traffic handling, fostering greater flexibility and responsiveness in ever-changing network conditions. However, traditional routing algorithms struggle to optimize different QoS constraints simultaneously in real-time network conditions, even with active (as opposed to reactive) decision points SDNs.

More recently, RL integrated DLT formed an area of interest to improve network management, especially from an adaptive and decentralized perspective. The thesis is concentrating on applying a DAG-dependent DLT and combining it using RL to improve the traffic management within SDN. The idea is to construct an optimal routing path discovery framework with QoS guarantees, and our framework must outperform the innovative method. This result offers many startup advantages using DAG-based DLT

against ancient blockchain technology—it is acyclic and non-linear structure accelerates the sound processing time of sections in a complex network. When integrated with RL's ability to dynamically adapt routing paths in response to real-time feedback from the network, it is expected that the framework will provide a more responsive and efficient way of traffic management. In this thesis, a comparative study will be performed between the traditional method of Dijkstra and OSPF with this new method, as well as DRM and PRM, on various performance parameters like FET, MEE, APL, ANoL and ANL.

1.2 Problem Definition

The rapid surge in data traffic has significantly heightened the need for networks that can accommodate diverse QoS requirements, such as bandwidth, latency, and packet loss. These challenges become even more pronounced across a wide array of devices and applications. However, traditional routing protocols like Dijkstra's and OSPF fall short in optimizing multiple QoS metrics simultaneously, especially in expansive and dynamically evolving environments.

In this context, DLT, specifically blockchain, has been explored as a tool for decentralized transaction validation. However, blockchain's linear processing model introduces scalability limitations, making it unsuitable for real-time, high-throughput networks like SDNs. DAGs offer an alternative approach, providing enhanced scalability and parallel transaction validation, making them more suitable for QoS-driven traffic management.

To formalize the traffic management problem in a multi-domain SDN environment, this study adopts the Inter-Network QoS-aware Routing Problem (IN-QRP) as described in Karakus (2021). The IN-QRP is defined mathematically as follows:

IN-QRP addresses the challenge of identifying QoS-compliant paths between ASes in a cross-domain SDN environment. The goal is to determine an optimal path from a source node in one AS to a destination node in another while ensuring compliance with QoS constraints, such as bandwidth availability, delay, and packet loss.

This problem is formally modeled as an overlay network $N(V, P)$ where each pathlet $p_i^j \in P$ is associated with a QoS weight vector $w_i(p_i^j) \geq 0$ for $1 \leq i \leq m$, where m represents the number of QoS parameters. Given a set of predefined constraints L_i for each metric, the IN-QRP problem aims to find a feasible path P between two border nodes $s \in* V_i$ and $d \in* V_j$, where $s \neq d$ and $* V_i \neq* V_j$, subject to:

$$\boldsymbol{w_i(p_i^j) \overset{\text{def}}{=} \bigodot_{(u,v)\epsilon p_i^j} w_i(u,v) \leq L_i, \qquad i = 1, \ldots, m} \tag{1}$$

where $\odot$ represents different operations depending on the nature of the constraints:

- Summation $(\sum)$for additive constraints (e.g., total delay)
- Product $(\prod)$for multiplicative constraints (e.g., reliability factor)
- Minimum $(\min)$for concave constraints (e.g., minimum available bandwidth)

A QoS-feasible path is any path that satisfies all mmm constraints. Since multiple such paths may exist in the overlay network $N(V, P)$, the goal is to select the optimal path based on additional objectives, such as cost minimization, load balancing, or network efficiency.

This formulation lays the foundation for adaptive traffic engineering in cross-domain SDN environments, where dynamic path selection must accommodate real-time QoS fluctuations. The proposed framework builds upon this model, integrating DAG-based DLT and RL to dynamically optimize inter-AS routing while ensuring scalability, robustness, and efficiency in large-scale, distributed networks.

This mathematical formulation encapsulates the core challenges of ensuring QoS compliance while optimizing network performance in multi-domain SDN environments. The lack of comprehensive models that integrate DAG-based DLTs with RL for QoS-driven routing further highlights the research gap addressed in this thesis.

While RL has demonstrated potential in optimizing real-time routing decisions in dynamic network conditions, its application in QoS-focused scenarios within SDNs remains underexplored. Existing RL models often address a single QoS parameter and fail to incorporate decentralized transaction validation mechanisms like DAG-based DLTs, which are essential for expandable and responsive traffic management.

Addressing the drawbacks of conventional methods such as Dijkstra's and OSPF, this thesis compares Decentralized Routing Method (DRM) and Pyramidal Routing Method (PRM) in evaluating the proposed framework. Key performance indicators (KPIs), including FET and MEE, are tailored to reflect QoS-specific requirements.

The proposed framework combines DAG-based DLT with RL to deliver an adaptive, scalable, and decentralized traffic management solution capable of maintaining QoS in distributed networks. By dynamically responding to changing network conditions, it addresses the complexity of large-scale traffic flows that traditional methods struggle to manage efficiently.

1.3 Contributions of the Thesis

This research significantly contributes to the fields of SDN, DLT, and RL by introducing a novel traffic management framework. Its primary impact lies in advancing traffic management strategies for modern networks. The key contributions of this study to traffic management in modern networks are as follows:

- **DAG-Based DLT Framework:** By introducing DAG as a decentralized alternative to traditional blockchain, this framework overcomes the bottlenecks of linear transaction processing. The non-linear architecture of DAG enables faster and more efficient transaction validation, making it a game-changer for high-throughput, real-time network scenarios.
- **RL for QoS Optimization:** Leveraging a Q-learning-based RL model, the study proposes a dynamic mechanism to optimize routing decisions in real-time. By continuously adapting to evolving network conditions and key QoS parameters

like bandwidth and delay, this approach ensures that routing decisions remain not only efficient but also highly responsive.

- **Development of a Scalable, Decentralized, and Adaptive QoS Traffic Management Framework:** The proposed DAG-based QoS-Centric traffic management framework (DQRL) framework introduces a decentralized architecture with adaptive capabilities that not only meet the intricate QoS requirements of modern networks but also provide a robust foundation for scalability and dynamic performance in next-generation network systems. By integrating DAG-based DLT with RL, DQRL ensures decentralized and adaptive traffic management, dynamically adjusting to network conditions to optimize performance and maintain QoS compliance in large-scale environments.

  Furthermore, DQRL comprises two specialized variants: DQRL_BW and DQRL_D. DQRL_BW is designed to optimize throughput for bandwidth-intensive traffic flows, ensuring efficient resource allocation and high data transmission rates in large-scale networks. Conversely, DQRL_D prioritizes low-latency routing, making it particularly suitable for real-time applications and time-sensitive communication services. By offering these tailored routing strategies, DQRL enhances flexibility and adaptability, providing a comprehensive solution that meets diverse QoS demands while ensuring scalability and efficiency in evolving network infrastructures.
- **Novel Routing Evaluation Approach:** The study benchmarks the proposed framework against DRM and PRM, moving beyond traditional algorithms like Dijkstra's and OSPF. Evaluation metrics such as FET, MEE, and APL, ANoL and ANL are used to measure the framework's performance.
- **Contribution to Traffic Engineering Research:** This study bridges a significant gap in traffic engineering by combining machine learning techniques with DAG-based DLT, offering a new perspective on adaptive and decentralized traffic management in SDNs.

1.4 Scope of the Research

In this thesis, a comprehensive traffic management framework is designed for SDNs using DAG-based DLTin conjunction with RL to enhance the path selection mechanism while ensuring QoS compliance.

This research initially centers on SDNs, highlighting their capability to decouple the control layer from the data layer. This decoupling allows for centralized management and programmability, paving the way for dynamic and effective traffic handling within networks. The study also examines how integrating decentralized technologies, such as DAG-based DLT, can enhance network optimization within SDN environments.

The second focus is the integration of DLT and DAG. This research emphasizes the use of DAG as a decentralized ledger within the SDN architecture. Unlike traditional blockchain, DAG provides greater scalability and faster transaction processing. The study details how DAG-based DLT can be utilized for decentralized transaction validation, contributing to more efficient traffic management within SDNs.

Another critical aspect of this research is the application of RL for routing. Specifically, it investigates the use of Q-learning to dynamically optimize routing paths in SDNs. This approach allows the system to adapt to real-time network conditions by continuously refining its strategies based on key QoS parameters, including bandwidth, latency, and delay.

The optimization of QoS is another critical component. This study aims to demonstrate how the proposed framework can enhance key QoS metrics, such as reducing latency, increasing bandwidth, and improving throughput, when compared to traditional routing protocols like Dijkstra's algorithm and OSPF.

Regarding the simulation and performance evaluation, this study uses Mininet for network emulation, Python for implementing the RL algorithm, and Matlab for

performance analysis. Various network topologies and traffic conditions will be simulated and tested, with the results being analyzed using KPIs such as FET, MEE, APL, ANoL and APL.

During the evaluation phase, the proposed framework's performance will be benchmarked against the DRM and PRM, following the methodology outlined in collaboration with my advisor's previous work. Evaluation metrics such as FET, MEE, APL, ANoL, ANL and total network throughput will be emphasized to reflect the specific performance benchmarks of QoS-sensitive environments.

A comparative analysis with traditional methods will be conducted, highlighting the strengths and limitations of the proposed DAG-based RL traffic management framework. This analysis aims to showcase how the new approach outperforms conventional methods, particularly in terms of scalability, adaptability, and QoS optimization.While the study is limited to simulated network environments, the simulations are designed to closely resemble real-world conditions, ensuring that the results are applicable to practical implementations. Although Q-learning is the primary focus of this study, future research may explore other RL techniques or hybrid models to further enhance performance.

1.5 Structure of the Thesis

This study develops a novel traffic management framework, DQRL, for SDNs by integrating DAG-based DLTwith RL. The research investigates how SDNs can leverage centralized control and programmability to manage dynamic network traffic more efficiently, while addressing the scalability limitations of traditional blockchain by utilizing DAG-based DLT for decentralized and fast transaction validation.

The study also focuses on the application of Q-learning to optimize routing paths in SDNs, based on real-time QoS metrics such as bandwidth, latency, and delay. The proposed framework dynamically adjusts to changing network conditions and improves network performance by ensuring compliance with QoS requirements. Simulations are

conducted in controlled environments using Mininet for emulation, Python for RL implementation, and Matlab for performance analysis. Various network topologies and traffic conditions are tested to validate the scalability and effectiveness of the proposed framework. KPIs such as FET, MEE, ANL, ANoL and APL are used to benchmark the results against established methods like DRM and PRM.

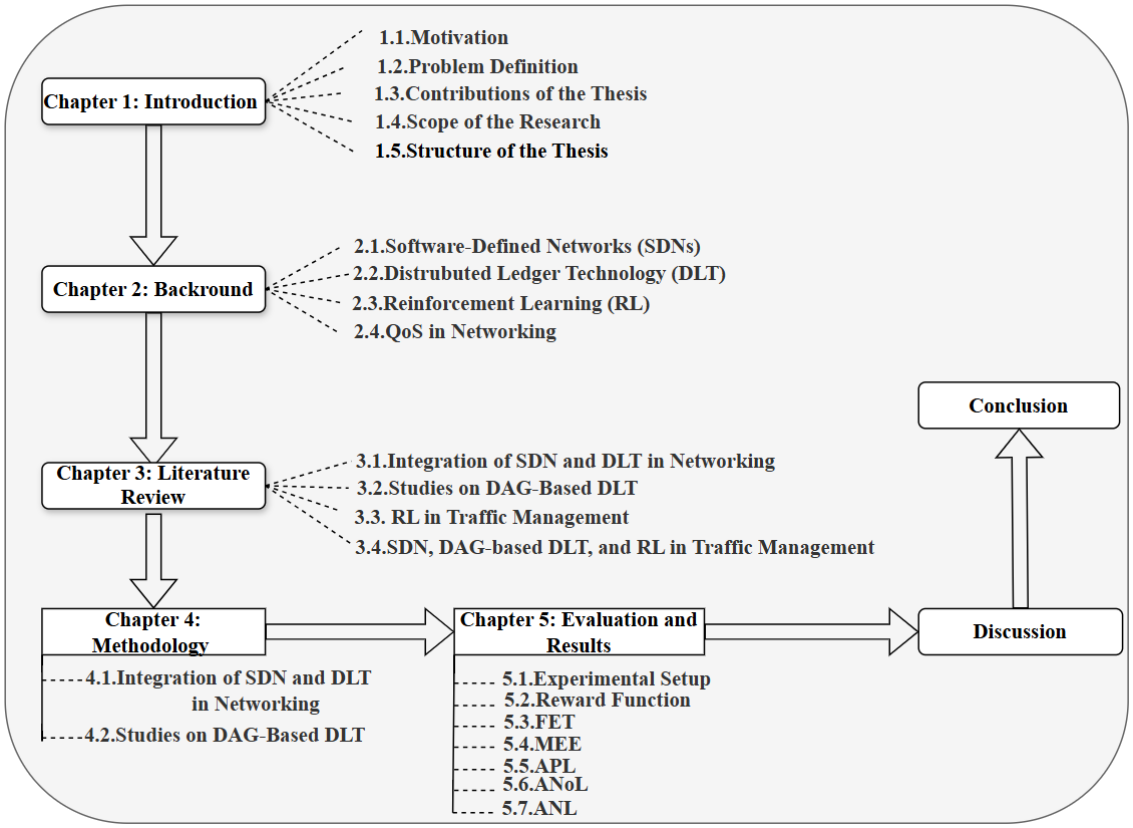


Figure 1.4 Structure of thesis diagram

Figure 1.4 illustrates the overall structure of this thesis, outlining the logical flow of the research. The thesis begins with Chapter 1: Introduction, which provides an overview of the research problem, objectives, and significance. Chapter 2: Background delves into the foundational concepts essential to this study, including SDNs, DLT, RL, and QoS in networking. Chapter 3: Literature Review analyzes existing work in the fields of SDNs, DLT, and RL, highlighting the research gaps addressed by this thesis. Chapter 4: Proposed Approach describes the novel DAG-based DLT and RL-integrated traffic management framework, detailing its design and functionality. Chapter 5: Methodology

outlines the experimental setup, simulation tools, and evaluation metrics employed to assess the framework. Chapter 6: Results presents the performance analysis and comparative evaluation of the proposed framework relative to existing approaches.

Following the results, the Discussion section interprets the findings within the broader context of network traffic management. It examines the advantages of integrating DAG-based DLT with RL, emphasizing improvements in scalability, adaptability, and QoS optimization. Additionally, it discusses the comparative performance of the proposed framework against traditional and blockchain-based routing approaches, identifying key strengths and limitations. The potential impact of the DQRL framework in emerging network paradigms, such as 6G and edge computing, is also considered.

Finally, the Conclusion section summarizes the key contributions of this research, reinforcing the significance of a decentralized and learning-driven approach for traffic management in SDNs. It highlights how the proposed framework effectively addresses the limitations of existing methods and lays the groundwork for future advancements. Furthermore, possible research directions are outlined, including enhancements to RL models, further refinement of DAG validation mechanisms, and real-world deployment studies to evaluate the feasibility of large-scale implementation.

The study is constrained to simulated network environments; however, the scenarios are meticulously designed to emulate real-world conditions. Future research could build upon this work by investigating alternative RL methodologies or hybrid models to further enhance the framework’s scalability and adaptability to evolving network demands.

2. BACKGROUND

In this thesis, we introduce a novel, software-oriented framework for DLTbased on DAG, and propose an innovative solution to fulfill the QoS requirements in network traffic management by ensuring interoperability among different SDNs. The DAG structure provides a secure and trustless data architecture to bolster security as well. With DAGs replacing traditional blockchains, data transaction-aware nodes exhibit a hierarchical nature, enabling complex network scenarios like SDNs to benefit from low-latency and effective data routing.

SDNs separate the control Layer from the data Layer, allowing for centralized oversight and decision-making. This separation facilitates dynamic and efficient traffic management, ensuring optimal use of network resources. The integration of RL enhances SDNs by enabling real-time adaptation to fluctuating network loads and conditions. RL agents in SDN environments can automatically adjust paths (e.g., bandwidth optimization, latency minimization, and load balancing) based on QoS requirements in response to fluctuating traffic conditions, thereby improving network responsiveness and efficiency (Song et al., 2023; Zhang et al., 2022).

Addressing scalability and verification challenges, DAG-based DLT frameworks in SDNs allow multiple previous transactions to be verified by each transaction node. Unlike the linear structure typical of blockchain, this approach reduces computational intensity. This integration benefits network traffic environments that require real-time data processing and support high fault tolerance, particularly useful for QoS enforcement across multiple domains. Combined with RL, DAG-based DLT frameworks provide a multi-layer solution encompassing dynamic learning, resilience, and fast transactions in network management.

Furthermore, when RL operates over a DAG-based DLT layer in SDNs, it aids in managing cross-network issues related to routing optimization across multiple network domains per QoS requirements. Within this architecture, RL algorithms (e.g., deep Q-learning) are used to identify and reinforce effective traffic patterns, allowing networks

to adjust in response to anticipated load fluctuations. This flexibility benefits traffic load changes, enabling operators to enforce policies that ensure service continuity (Wang et al., 2021; Liu et al., 2023).

2.1 Software-Defined Networks (SDNs)

SDNs revolutionize conventional networking by separating the control Layer from the data layer, enabling centralized management and improved programmability. Unlike conventional architectures where control and data processes coexist in individual devices, SDNs consolidate decision-making within a centralized controller, enabling a holistic view of the network and streamlining management tasks. This architecture facilitates adaptive, demand-driven traffic flows and the rapid deployment of new services (Bilal & Khan, 2019).

The SDN architecture is structured into three primary layers: the infrastructure layer, which handles data transmission via network devices like switches and routers; the control layer, managed by an SDN controller to oversee flow management; and the application layer, which facilitates interactive control and customization through programmatic interfaces. This layered approach enhances resource allocation and traffic management while allowing for real-time policy adjustments (Akhunzada et al., 2016).

The programmability of SDN enables dynamic load balancing and network customization, making it ideal for environments like data centers and enterprise networks. For instance, Google employs SDN to manage its global data centers, dynamically optimizing routes based on network conditions (Agarwal et al., 2013).

Despite its advantages, SDN faces security challenges due to its centralized control. Vulnerabilities like Denial of Service (DoS) controller attacks and eavesdropping threats can disrupt network operations. Integrating firewalls and Intrusion Detection Systems (IDS) within the SDN controller has been proposed to mitigate such risks effectively (Ahmad et al., 2015).

SDN's applications extend to emerging fields like intelligent transportation systems, where real-time policy adjustments improve resource utilization and QoS management, demonstrating its versatility in diverse networking scenarios (Meneguette, 2020).

2.1.1 SDN architecture

SDN is a modern networking framework that emphasizes the separation of the control layer from the data layer. This approach supports a centralized management structure, enabling flexible and programmable control of network operations. Traditional networks combine control and data functions within individual devices, where they handle both packet processing and policy enforcement. In contrast, SDN separates these roles by delegating control tasks to an external, centralized controller. This controller manages packet flow across the network, while the data Layer is dedicated solely to delivering packets according to predefined rules issued by the controller. This architecture can operate alongside or independently of proprietary software on hardware switches, providing greater adaptability and efficiency.

This separation helps the SDN controller observe the network globally and make globally informed, performance-efficient decisions that optimize resource utilization (Faizullah & Almutairi, 2018).

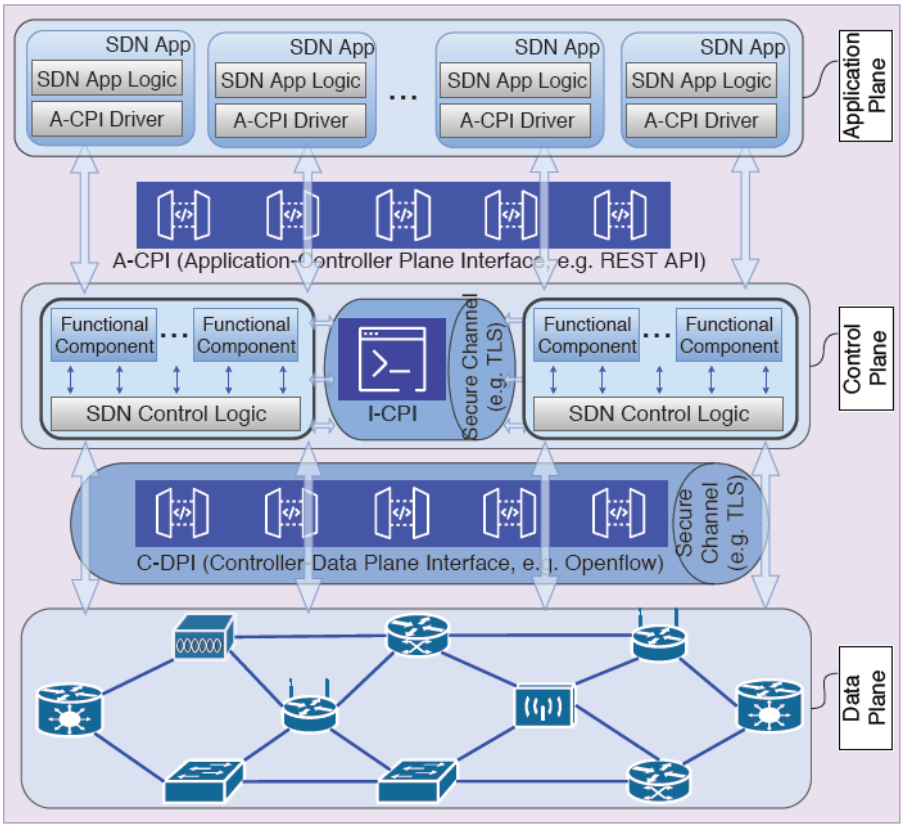


Figure 2.1 Overview of SDN architecture with DAG integration (Karakus, 2021)

Figure 2.1 provides an overview of the SDN framework, emphasizing the separation of the control layer and the Data layer. The SDN controller centrally manages the control Layer, establishing communication with the Data Layer via secure protocols, such as OpenFlow. Above the control layer, the application layer provides programmable interfaces to enable customization of network functionalities. In this thesis, the conventional blockchain-based approach depicted in Karakus (2021) has been revised to integrate DAG technology. The DAG-based architecture ensures enhanced scalability and faster transaction validation compared to traditional blockchain methods, making it more suitable for real-time SDN operations.

The SDN control Layer functions as the network's brain, making decisions about how traffic flows across all devices. Using OpenFlow, the SDN controller communicates with Data layer devices, allowing for dynamic updates to flow rules and adjustments to packet processing paths as required. By centralizing network intelligence, SDN significantly simplifies management tasks like traffic engineering and security policy enforcement,

resolving common issues of complexity and inflexibility associated with legacy networking approaches (Zhao & Wu, 2017). However, consolidating control intelligence can also introduce new vulnerabilities— however, disruptions in the control Layer can potentially impact the performance of the entire network. This challenge has prompted the development of distributed controller models designed to enhance SDN scalability and reliability (Gray et al., 2016).

The Data Layer's primary function is to handle the forwarding of data packets, operating directly at the device level to ensure efficient packet delivery. It performs flow rules defined by the control Layer and handles their execution, thus avoiding policy decision overhead and yielding efficient packet processing. This separation streamlines data Layer operations, focusing it on the rapid, efficient forwarding of data at network speeds with low latency. Additionally, this decoupling enables data Layers to be optimized for pure data forwarding, contributing to enhanced throughput and scalability across large network infrastructures (Yang et al., 2017).

Table 2.1 SDN architecture components

Component	**Function**	**Role in SDN Architecture**	**Advantages**	**Challenges**
Control Layer	Manages routing, traffic engineering, and policy enforcement.	Acts as the central decision-making layer, defining data flow.	Simplifies operations, enables scalability, and dynamic updates.	May create a single point of failure
Data Layer	Forwards packets based on rules from the Control Layer.	Operates as the forwarding layer, ensuring data transmission.	High-speed forwarding, low complexity, and direct execution.	Fully dependent on Control Layer, increasing latency under heavy load.
SDN Controller	Executes network-wide policies and oversees forwarding devices.	Central intelligence, coordinating between layers for seamless operation.	Supports programmability, centralized policy enforcement, and real-time adjustments.	Scalability issues
Application Layer	Provides advanced functions like QoS, security, and traffic optimization.	Communicates with the SDN Controller via APIs to enhance functionality.	Enhances flexibility, enabling custom applications for specific needs.	Complex integration

Table 2.1 offers a detailed perspective on the fundamental components of SDN architecture, highlighting their roles, benefits, and limitations within the network ecosystem. The Control Layer serves as the intelligence hub of SDN, consolidating network policies and directing traffic flows. While this centralized approach streamlines policy enforcement and offers superior network oversight, it introduces potential risks, especially in single-controller setups where failover mechanisms become crucial to ensure resilience. The Data Layer, by contrast, performs straightforward packet forwarding based on predefined flow rules, minimizing computational complexity and ensuring swift data handling. However, its dependency on the Control Layer limits its responsiveness, as it lacks decision-making capabilities and may face delays awaiting flow rules. The SDN Controller functions as the network's primary decision-making entity, synthesizing policy directives from the Application Layer and relaying instructions to Data Layer devices. Its centralized nature enhances programmability and dynamic reconfiguration but can lead to scalability challenges in extensive network deployments, necessitating distributed controller configurations. Finally, Network Applications provide customizable services through APIs, enabling tailored functionalities such as QoS management and security enforcement. Although this modularity fosters innovation, the dependency on well-defined APIs can sometimes complicate integration, especially in complex or heterogeneous network environments. Together, these components establish a robust yet adaptable architecture, supporting SDN’s goals of flexibility, central control, and enhanced functionality in dynamic network landscapes (Arora et al., 2018).

This division between control and Data Layer functions is crucial for enabling programmable network environments. By allowing the SDN controller to dynamically adjust network policies in real time, organizations can respond more flexibly to changing needs and optimize resource use. This capability is particularly beneficial for data centers and cloud environments, where the ability to quickly scale processing in response to traffic pattern changes is critical (Carvalho et al., 2020). The SDN Controller operates as the core decision-making entity, bridging the Application Layer's policy inputs with the Data Layer's execution of these directives. By segregating the control layer from the data layer, SDN architecture enables dynamic, efficient, and easily manageable network operations.

2.1.1.1 Data layer

The Data Layer is the backbone of SDN. It's the layer that actually moves data packets around, following rules set by the Control Layer. Think of it as the muscle of SDN—carrying out the decisions made by the brains, which is the Control Layer. This layer includes network devices like switches and routers that take orders from the SDN controller. Unlike traditional networks, where each device makes its own decisions about where to send data, the Data Layer in SDN is more of an executor, ensuring consistent, policy-based traffic flow across the network.

In Data Layer, devices act as "packet movers" without needing complicated routing logic or setups. Each device has a flow table filled with rules from the SDN controller. These rules tell the device what to do with each packet it receives—whether to forward, drop, or modify it—based on details like IP addresses or protocol types (Zhao & Wu, 2017). When a packet is received, the device examines its flow table to locate the appropriate rule, and takes the specified action.

Data Layer's core tasks can be grouped into several main functions:

- **Packet Forwarding**: This is the Data Layer's most basic job. Every packet gets sent through the network according to pre-set flow rules. Packet forwarding is key for smooth traffic flow, especially in networks with high data loads, because it allows data to travel quickly and reliably. This process is fine-tuned to reduce latency, so packets reach their destinations without unnecessary delays (Akhunzada et al., 2016).
- **Flow Rule Matching**: When a packet hits a Data Layer device, the device uses its flow table to figure out how to handle it. These rules map out each packet's route through the network, allowing SDN to manage traffic flow based on the network's larger goals—like balancing the load, ensuring QoS, or enforcing security policies (Bilal & Khan, 2019).
- **Traffic Isolation and Prioritization**: SDN's Data Layer also supports more refined traffic management. It can isolate certain types of data or prioritize some

flows over others, depending on factors like source, destination, or application needs. For example, time-sensitive traffic like video calls or VoIP can get priority over less urgent data like file backups. This ability to control traffic at a detailed level helps SDNs provide better service for different applications and users (Meneguette, 2020).

- **Packet Processing Capabilities**: Some advanced Data Layer devices can make changes to packets on the fly, adding tags, encapsulating, or decapsulating them as needed. These processing capabilities help make sure packets are properly identified and handled across different parts of the network. Packet processing also supports QoS by ensuring that traffic meets necessary standards for latency, bandwidth, and reliability (Agarwal et al., 2013).
- **Monitoring and Reporting**: Although Data Layer devices don't make independent control decisions, they still play a big role in tracking network traffic. They can log details about packet flows, like how many packets were transmitted, dropped, or modified, and send this information back to the Control Layer. This data helps the SDN controller monitor network performance and adjust routing as needed. Monitoring is essential for maintaining network health and enabling data-driven decision-making by the controller.

The Data Layer's design emphasizes speed and reliability, reducing the need for complex configuration on each device. By shifting decision-making to the Control Layer, the Data Layer can focus purely on forwarding data, which lightens the load on individual devices and improves scalability. This setup also makes network behavior more predictable, as the Data Layer sticks closely to the policies set by the controller. This streamlined approach is especially beneficial in large networks, where it's tough to ensure consistent performance across thousands of devices using traditional setups (Faizullah & Almutairi, 2018).

2.1.1.2 Control layer

In SDN, the Control Layer serves as the primary decision-making component, governing network operations by defining the rules and policies that the Data Layer executes for

traffic management. Unlike conventional networks, where individual devices independently process forwarding decisions using embedded control functions, SDN decouples the Control Layer from physical devices and centralizes it within an SDN controller. This centralized design grants the controller a unified and comprehensive network view, facilitating streamlined and coordinated management of data traffic.

The Control Layer acts as a central command hub for the whole network. It controls traffic flow, resource allocation, and security policies from one place, allowing administrators to make changes that instantly apply network-wide. This centralized approach cuts down on configuration inconsistencies and reduces the workload for IT teams (Kreutz et al., 2015). It also makes the network more flexible, letting it quickly adapt to changes like traffic spikes or resource adjustments in real time.

A major role of the Control Layer is to set and enforce policies that keep the network running smoothly, securely, and with the necessary QoS. Policies might include load balancing, managing bandwidth, or controlling access based on user or application needs. With its bird's-eye view of the network, the Control Layer can make smart, real-time decisions to use resources effectively, which is especially helpful in fast-paced environments like data centers or cloud networks (Birkner et al., 2016).

The Control Layer establishes flow rules that guide the Data Layer in handling packets, specifying actions such as forwarding, dropping, or modifying them as they move through the network. These rules are dynamically updated by the SDN controller to accommodate real-time changes in network conditions, allowing the Control Layer to adapt traffic patterns to meet demand. By distributing these instructions to Data Layer devices, the Control Layer ensures that the network operates efficiently and aligns with performance objectives (McKeown et al., 2008).

The Control Layer keeps a close watch on network activity, gathering real-time data on traffic patterns, device performance, and any unusual events. This data is essential for analytics and troubleshooting, allowing the controller to detect issues—like congestion or security risks—and take action to fix them. For example, if there’s a bottleneck, the

Control Layer can reroute traffic to keep things running smoothly. The insights from monitoring also help refine policies over time, making the network smarter and more efficient (Katta et al., 2016).

The SDN controller automates many configuration tasks, so changes can be made quickly without needing manual updates on each device. This automation is especially useful in environments where the network has to scale or change frequently. For example, in dynamic environments like cloud data centers, where virtual machines are frequently added or removed, the Control Layer's automation allows the network to quickly adjust to these changes, facilitating seamless operations (Jain et al., 2013).

The Control Layer also connects the Data Layer to the Application Layer, allowing apps to interact with the network. Through APIs, applications can request specific network services or apply policies tailored to their needs, like prioritizing traffic for video calls. This interaction lets the Application Layer communicate directly with the network infrastructure, making the network more responsive to app-specific demands (Shin et al., 2014).

Centralizing the Control Layer offers significant advantages, such as enhanced network visibility, simplified management, and increased flexibility. However, this approach also introduces challenges, particularly concerning security and reliability. A compromised SDN controller, as the central point of control, could jeopardize the complete infrastructure. To mitigate this risk, advanced SDN architectures employ redundancy mechanisms, such as multiple controllers or distributed control Layers, to improve network resilience (Heller et al., 2012).

2.1.1.3 Application layer

In SDN, the Application Layer acts as the interface where advanced network applications interact with the SDN controller to establish and influence network operations. This layer connects the network's infrastructure, comprising the Data and Control Layers, with

applications that require specific functionalities, allowing developers to tailor the network's behavior to suit varying requirements. Through APIs, the Application Layer interacts with the Control Layer, enabling applications to manage aspects such as traffic routing, load balancing, security measures, and QoS, dynamically adapting to real-time demands.

One major role of the Application Layer is traffic engineering—optimizing data flow across the network to meet specific performance goals. Traffic engineering apps can use SDN's centralized control to adjust routes, manage congestion, and make sure resources are used efficiently. For example, during peak times, traffic can be rerouted to avoid bottlenecks, keeping data flowing smoothly and efficiently (Mendiola et al., 2017).

QoS applications on the Application Layer can prioritize certain traffic types to ensure high performance for services that need it, like video streaming or online gaming. By working with the SDN controller, QoS apps can adjust bandwidth, delay, and jitter based on service level agreements (SLAs) or real-time network conditions. This helps service providers guarantee good performance for critical applications and deliver a consistent experience to users (Guerzoni et al., 2014).

Security applications on the Application Layer are essential for enforcing security policies, spotting unusual activity, and blocking unauthorized access. These apps can set up firewall rules, IDSes, and, Access Control Lists (ACLs) directly through the SDN controller, offering a centralized approach to security. By analyzing traffic and spotting potential threats, security apps can update policies in real time to mitigate risks, making the network more resilient against attacks (Scott-Hayward et al., 2015).

Load balancing apps on the Application Layer help spread traffic evenly across network resources, preventing any one path or server from getting overwhelmed. This is especially important in data centers or cloud networks, where demands can shift rapidly. By working with the SDN controller, these apps can manage traffic distribution in real time, improving reliability and reducing latency by keeping resources well-balanced (Xie et al., 2016).

The Application Layer also hosts analytics and monitoring apps, providing administrators with insights into performance, usage patterns, and potential issues. These apps can collect data on metrics like bandwidth usage, packet loss, and latency, supporting data-driven decision-making. With these monitoring tools, administrators can spot performance dips or problems early, enabling quick fixes and ongoing optimization of the network (Ayoubi et al., 2018).

Policy management apps let administrators enforce rules based on organizational needs, compliance standards, or business objectives. For instance, they might set policies to limit access to certain apps at specific times or prioritize critical traffic over non-essential traffic. These applications collaborate with the SDN controller to convert high-level policies into actionable flow rules, ensuring efficient data movement and seamless enforcement of network policies. This integration helps align network behavior with organizational objectives, streamlining operations and enhancing adaptability (Li & Chen, 2017).

The Application Layer's programmability makes it highly flexible, allowing for custom applications that meet unique network needs. This layer transforms the network from a static system into a dynamic platform that adapts to real-time demands. Developers can use open APIs to interact with the SDN controller, creating solutions that respond to emerging requirements. This flexibility is particularly useful in cloud environments, where demand is unpredictable, and applications need to adjust quickly to changes in network load, user activity, or security threats (Kreutz et al., 2015).

In essence, the Application Layer is the innovation center of SDN, where application intelligence meets network programmability to build smarter, more adaptable, and more secure networks. By enabling advanced applications, the Application Layer allows organizations to fine-tune their networks for the unique needs of their users and applications, leading to major gains in efficiency, security, and overall user experience.

2.1.2 Fundamental components of SDN

As for SDN, the most important parts are SDN controllers, switches, and network applications. Working together, these components are designed to bring about the flexible and programmable network management features that SDN can deliver.

The SDN controller serves as the primary decision-making unit, often referred to acts as the core decision-making unit of the network. Controllers interpret network policies and communicate with each switch in the network to manage data flows. This centralized management provides a cohesive network-wide view, enabling advanced functions like load balancing, traffic engineering, and security policy enforcement. Controller architectures may vary, with some systems incorporating distributed or multi-controller models to improve scalability and resilience, minimizing the risk of a centralized point of failure (Arora, 2018). Load balancing among multiple controllers in dynamic networks helps optimize performance by adjusting traffic distribution according to network demands (Sufiev et al., 2019).

SDN Switches work at the data layer level, managing packet transmission in accordance with controller instructions. By design, SDN switches are reactive and do not independently forward packets but rely on flow rules from the controller, which specify packet handling. Separating control and forwarding simplifies switch operation, allowing switches to focus on packet processing. OpenFlow, one of the most widely used protocols for controller-switch communication, supports basic match-action functions to set packet flows in the network (Li et al., 2018). Switches are integral to SDN operations, as they implement the policies issued by the controller in real-time. However, this reliance on the controller's instructions can sometimes introduce latency and affect overall performance (He et al., 2015).

In multi-controller SDN setups, Inter-Controller Communication Protocols are key for keeping controllers working together smoothly. These protocols help controllers share data and stay in sync, so they can exchange network state info, routing decisions, and policy updates in real-time. Technologies such as OpenFlow, Border Gateway Protocol

(BGP), and East/West APIs are commonly utilized to maintain an updated view of the network's structure and traffic structures for the controllers. By using effective inter-controller protocols, networks can perform better, balance traffic loads across controllers, and lower the risk of network splits. This coordination becomes especially important in large or geographically spread-out networks, where a single controller might struggle to handle all the traffic on its own (Zhang et al., 2020).

Network Applications in SDN use the controller's programmability to deliver specialized network services like security monitoring, intrusion detection, and optimized routing. Applications communicate with the controller through northbound APIs, allowing network administrators to manage and customize functionalities without requiring modifications to the underlying hardware. By hiding network complexity, SDN applications optimize resource utilization and provide tailored service quality in multi-tenant environments like data centers. Component-based frameworks, such as CFCC, facilitate easy application development by allowing developers to integrate and reuse network functions as modular components (Qi & Shen, 2017). This modular approach enhances SDN application scalability and flexibility, supporting rapid adaptation to changing network needs.

2.1.3 Key advantages of SDN

SDN offers significant benefits, including centralized control, enhanced flexibility, and improved scalability, revolutionizing traditional network architectures and increasing operational efficiency.

A key benefit of SDN is its centralized management. By consolidating network control into a single or logically centralized controller, administrators can implement and oversee policies across the entire network from one location. This streamlines tasks like traffic engineering, load balancing, and enforcing security policies. Centralized management also enhances visibility across the network, providing real-time insights and enabling swift responses to changes—a crucial feature in dynamic, large-scale environments (Sallam et al., 2019). Moreover, it improves QoS management by allowing precise

monitoring and control of traffic flows to meet specific performance criteria (Nguyen et al., 2016).

The flexibility of SDN comes from its programmable architecture, which allows for dynamic adjustments to network configurations and policies as demands change. This programmability fosters innovation in deploying new services since network functions can be tailored to specific application requirements without modifying the physical infrastructure. This adaptability is especially valuable in environments like cloud data centers and IoT networks, where resource needs frequently shift. Additionally, SDN's programmability enables the integration of custom applications—such as security protocols and monitoring tools—enhancing its functional adaptability even further (Bassene & Gueye, 2022). In terms of security, this flexibility allows the controller to implement threat detection and mitigation policies in real-time across the network, adapting to evolving cybersecurity threats (Chica et al., 2020).

One big advantage of SDN is that it's cost-efficient. By segregating the control layer from the data layer, SDN minimizes the dependency on specialized hardware, streamlining network operations. letting organizations rely on cheaper, off-the-shelf switches and routers instead. This setup cuts down Capital Expenses (CAPEX) by reducing reliance on proprietary equipment, and it lowers Operational Expenses (OPEX) thanks to simpler network management. SDN's programmability also makes configuring and maintaining the network less expensive and complex, since changes can be rolled out centrally across devices without manual tweaks. These savings are particularly appealing to organizations that manage large networks or data centers (Kim et al., 2020).

SDN also gives administrators fine-grained control over network traffic. With centralized control, SDN can apply specific flow rules based on the needs of each application, user role, or security policy—something traditional networks can't easily match. This kind of precision enables use cases like prioritizing latency-sensitive apps, isolating suspicious traffic, or setting bandwidth limits per flow. Such detailed traffic control helps optimize QoS and improve resource use, which is especially useful in demanding environments

like multi-tenant data centers or IoT networks with diverse application needs (Bannour et al., 2018).

Scalability is a fundamental strength of SDN, enabled by its structure that divides the control and data layers, allowing networks to grow and adapt seamlessly. Traditional networks often struggle with scalability due to limitations in distributed control mechanisms. In contrast, SDN supports both centralized and distributed control configurations. Deploying distributed controllers enables SDN to scale across vast geographical areas or networks with high device density without significant performance loss. By distributing the control load among multiple controllers, SDN manages increased data flows and network complexity more efficiently (Alsaeedi et al., 2019). For example, hierarchical controller architectures can reduce latency and processing overhead by segmenting the network into manageable regions, allowing SDN to maintain low-latency performance even in extensive networks (Azodolmolky et al., 2013).

2.1.4 Fundamental challenges in SDN

Software Defined Network or SDN for short stats are a world of benefits and also some interesting security performance & interoperability issues. This directly affects its widespread implementation and usage in complicated network setups.

Despite all these benefits, the main issue associated with SDN is security because since the initiation of a single controller / control Layer that can further be considered as a new one point p) failure. Digital attacks like Distributed Denial of Service (DDoS) attacks and eavesdropping-based threats, and stream rule control can mishandle SDN vulnerabilities to disrupt network activities. More effective targeted attacks can be launched against the centralized controller, of which when successfully compromised grant control over the entire network. Strategies to counter these security concerns include sophisticated controller placement, and the use of dedicated protocols such as OpenFlow for network flow rule policies security. Yet we still have an ongoing requirement for a highly resilient security posture that can more dynamically react to threats in the SDN architecture (Yang et al., 2020; Hegazy & El-Aasser, 2021).

From a performance perspective, the centralization of control Layer in SDN could induce bottlenecks with increasing network footprint/complexity. It results in the performance degradancy like High Latency, Poor Throughput and Controller becoming bottleneck as controller has to process large size of Network traffic & requests. There are different solutions suggested by researchers to enact oncontroller, distributed controller frameworks that help in reducing the load of a single controller andas well as improves response time from controllers for high-traffic environments. On the other hand, efforts to optimize SDN controller performance with algorithms and load-balancing methods have been reported effective in reducing flow latency and improving global throughput for large-scale SDNs (Sokappadu et al., 2019; Islam et al., 2020).

Interoperability is yet an another significant obstacle; mostly, SDN environments are integrated with legacy network architectures and multivendor devices. Diversity in implementations by different vendors and inconsistencies in protocols may prevent easy integration of SDN with existing network components and may well lead to some incompatibility issues. To resolve such interoperability issues, an interoperability framework in association with standardized protocols is essentially needed to provide assurance that controllers and switches from different vendors can communicate well with a variety of network devices. It has also been proposed that hybrid architecture combines protocols enabled by SDN with traditional networking protocols so that the transition of networks to SDN will be done gradually while maintaining better compatibility at no significant functional costs (Dixit et al., 2018; Scott-Hayward et al., 2016).

2.2 DLT

DLThas revolutionized data management, with blockchain being its foundational model. However, blockchain's linear structure faces limitations in scalability and transaction throughput. DAG have emerged as a more flexible alternative, enabling multiple transactions to occur simultaneously through a multi-directional graph structure. This approach enhances scalability and reduces latency, making DAG-based systems suitable

for high-speed applications such as IoT, gaming, and smart grid systems (Benčić & Žarko, 2018; Mönch & Rizk, 2023).

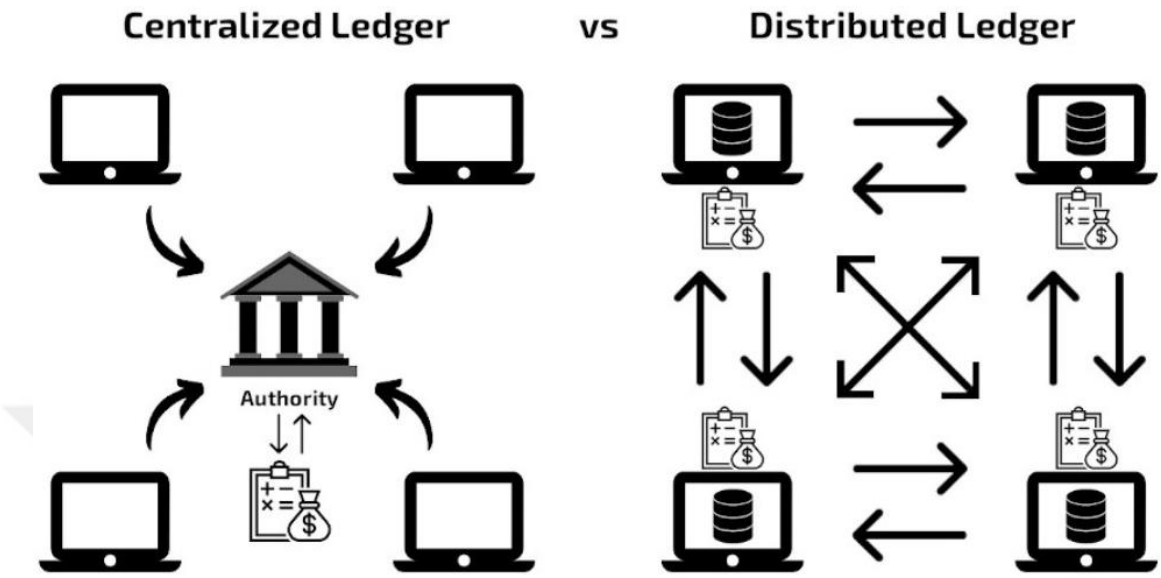


Figure 2.2 Comparison of centralized and decentralized ledger systems

DAG systems eliminate the reliance on energy-intensive consensus mechanisms like Proof of Work (PoW), offering a transaction validation process that is both efficient and sustainable. Unlike traditional blockchain systems, which operate as a linear sequence of blocks validated through PoW or Proof of Stake (PoS), DAG enables decentralized transaction validation without mining, leading to faster processing and reduced energy consumption. These systems also reduce network congestion by providing continuous consensus, which is particularly advantageous in scenarios requiring instant confirmation, such as IoT environments (Fan et al., 2021).

Despite their advantages, DAG-based DLTs face unique security challenges, including the "parasite chain attack," which exploits the decentralized structure to disrupt transaction validation. Researchers are exploring enhanced consensus algorithms and attachment mechanisms to address these vulnerabilities and improve the resilience of DAG systems (Cullen et al., 2020).

DAGs also provide superior fault tolerance compared to linear blockchain structures. By distributing transaction validation points, they mitigate the risk of a central failure point, making them an optimal choice for sectors such as healthcare and logistics management, where real-time data verification and system reliability are critical (Prostov et al., 2021).

2.2.1 Blockchain

Blockchain is the foundational model of DLT, introduced alongside Bitcoin in 2008 to enable decentralized transaction management without a central authority. It consists of a linear sequence of blocks, each cryptographically linked to its predecessor, ensuring data immutability and transparency. This structure is ideal for applications requiring trustworthy records, such as financial transactions, identity management, and supply chain monitoring (Nakamoto, 2008; Yli-Huumo et al., 2016).

In blockchain networks, nodes maintain a complete copy of the ledger, and transactions are validated using consensus protocols such as PoW or PoS. PoW ensures robust security by requiring significant computational effort, but its resource-intensive nature has led to the exploration of alternatives like PoS, which prioritize efficiency and environmental sustainability (Zheng et al., 2017; Xiao et al., 2020).

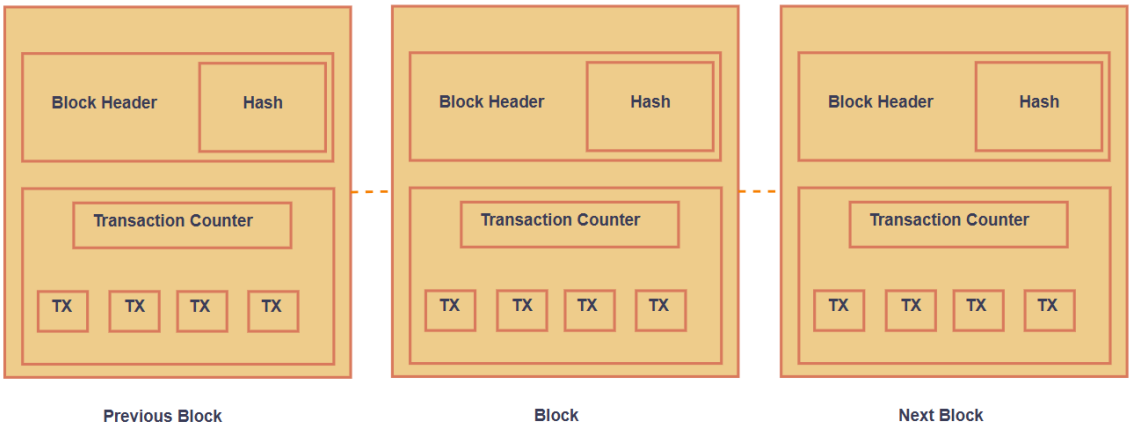


Figure 2.3 Blockchain structure overview

Figure 2.3 illustrates the basic structure of blockchain, consisting of sequentially linked blocks. Each block contains a header with metadata (e.g., previous block hash, timestamp) and a transaction section, highlighting the linear and immutable nature of the blockchain.

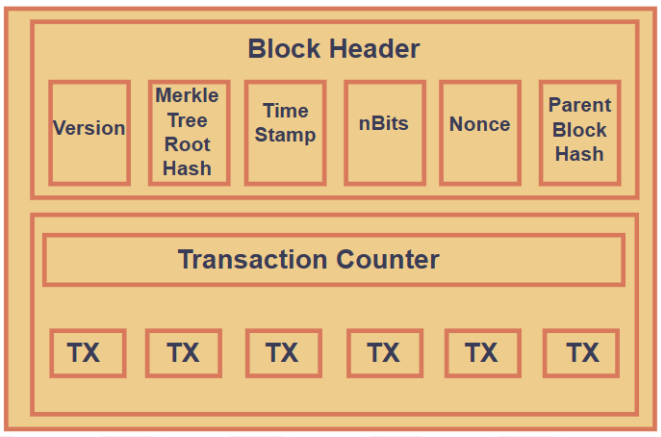


Figure 2.4 Detailed block structure

Figure 2.4 illustrates the structure of a single blockchain block, where the block header includes critical components such as the version, Merkle Tree Root Hash, timestamp, nonce, and parent block hash. The block body contains the transaction counter and transaction data, showcasing the organization and integrity of blockchain information.

Despite its advantages, blockchain's linear structure limits scalability and transaction speed. Each transaction must be processed sequentially, causing bottlenecks in high-demand environments like IoT or high-frequency trading. These limitations have prompted the development of alternative DLT models like DAG, which enable parallel transaction processing, addressing blockchain's scalability and energy concerns (Benčić & Žarko, 2018)

2.2.2 DAG

DAG is an advanced approach to DLTthat addresses scalability and latency issues associated with traditional blockchains. Unlike blockchain’s linear chain structure, DAG operates as a web-like graph where each transaction references multiple previous

transactions. This multi-directional design allows for parallel transaction validation, making DAG highly scalable and suitable for applications requiring real-time processing, such as IoT, financial services, and supply chain systems (Benčić & Žarko, 2018).

In DAG-based systems, there are no blocks or miners; instead, transactions validate each other directly. This eliminates the need for energy-intensive mining processes, enabling faster and more energy-efficient transaction processing. As network activity increases, DAG systems benefit from enhanced scalability, as each transaction contributes to validating others, reducing congestion and speeding up processing times (Popov, 2017; Fan et al., 2021).

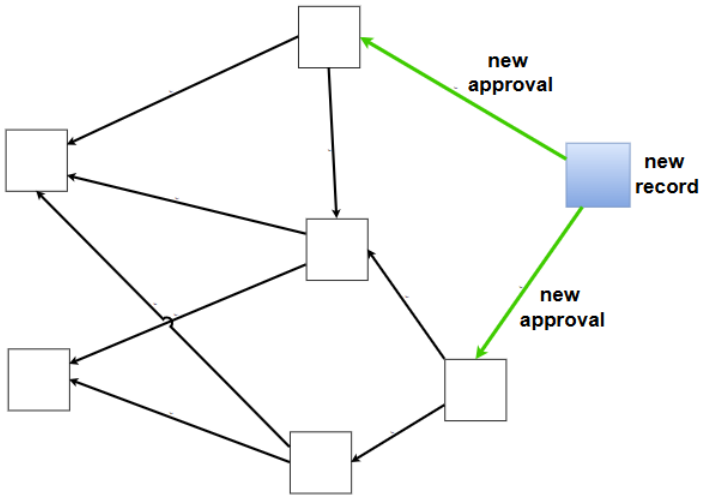


Figure 2.5 DAG-based Blockchain structure

Figure 2.5 depicts the DAG structure, a non-linear model that provides an efficient alternative to traditional blockchain. Figure 9 further demonstrates DAG's transaction validation process, where new records (blue nodes) reference and approve multiple previous transactions (green arrows). This decentralized validation eliminates the need for mining, enabling high transaction throughput and scalability by allowing concurrent transaction processing. Unlike blockchain, where transactions are sequentially linked, DAG allows transactions to reference multiple previous transactions, forming a web-like

structure. This design enables parallel processing, boosting scalability and reducing latency, particularly in high-transaction environments like IoT and financial systems.

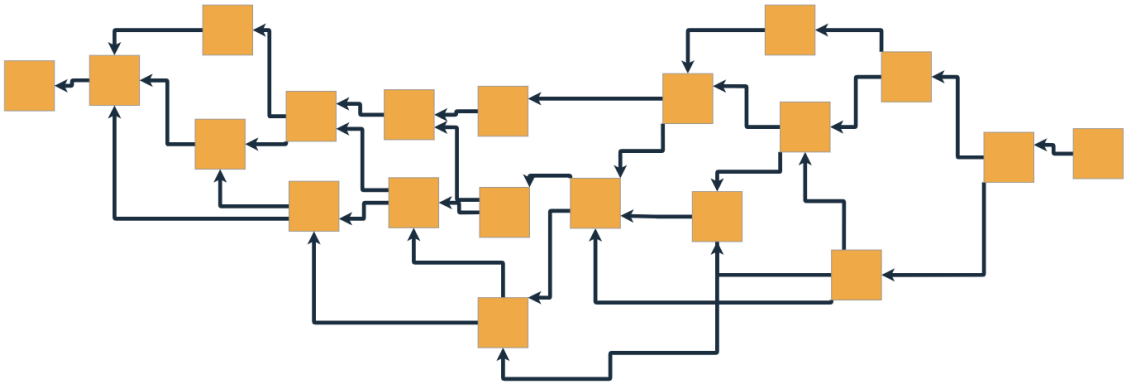


Figure 2.6 DAG transaction validation process

Figure 2.6 depicts the transaction validation process in a DAG system. In this approach, every new transaction references and approves several preceding transactions. This decentralized method removes the need for traditional mining or consensus mechanisms, such as PoW, resulting in faster and more energy-efficient validation processes.

DAG's decentralized architecture spreads transaction validation across all nodes, avoiding single points of failure and enhancing network resilience. This makes DAG ideal for critical applications in healthcare, finance, and smart grids that require continuous uptime and fault tolerance. However, DAG systems also face unique security risks, such as the "parasite chain" attack, which disrupts transaction integrity. Ongoing research focuses on developing advanced consensus mechanisms and linking strategies to address these vulnerabilities (Cullen et al., 2020).

DAG presents a scalable, energy-efficient, and eco-friendly alternative to blockchain for decentralized networks. While it offers significant advantages in transaction speed and energy efficiency, security challenges unique to its structure remain a key area of research.

2.2.3 Blockchain vs. DAG

DLT introduces modern methods for handling and verifying digital transactions across decentralized systems. traditionally, blockchain technology has served as the foundation of DLT with the aid of organizing transactions sequentially within blocks, every established with the aid of a consensus protocol which include evidence of work. This layout, however, encounters scalability and overall performance limitations, mainly as transaction quantity increases. To deal with those challenges, DAG have emerged as a promising opportunity, presenting a non-linear, multi-course method to transaction validation that notably complements transaction throughput and decreases latency (Pervez et al., 2018).

Unlike blockchain, which is based on a linear series of blocks, DAG organizes transactions in a web-like shape where every transaction is connected to preceding transactions without forming a non-stop chain. This shape permits DAG-based totally DLTs to method multiple transactions concurrently, bypassing the congestion and delay troubles typically associated with blockchains. DAG generation, as exemplified via platforms like IOTA and Nano, removes the want for miners via imposing a peer-validation version, where each new transaction confirms previous ones. This mechanism now not handiest enhances speed but also minimizes power consumption, making DAG-primarily based systems more sustainable and green (Dong et al., 2019).

At the same time as both blockchain and DAGs intention to ensure records integrity and security, DAG-primarily based DLTs provide wonderful advantages in IoT and actual-time facts programs due to their asynchronous transaction validation. This technique is in particular beneficial for environments requiring high-frequency micro-transactions, as seen in IoT networks, where latency and transaction fees are important concerns. in contrast to blockchains, DAGs do not require blocks to be mined, accordingly putting off transaction prices and taking into account a more seamless interaction between IoT devices (Watanabe & Shoji, 2023).

Table 2.2 Comparative analysis of Blockchain and DAG technologies

Attribute	Blockchain	DAG
Architecture	Sequential blocks storing multiple transactions.	Graph-based structure with interlinked transactions.
Validation Method	Uses PoW or PoS for block confirmation.	Transactions validate each other, removing the need for traditional consensus.
Throughput	Slower due to block mining constraints.	Higher due to parallel transaction processing.
Scalability	Limited by increasing transaction loads.	Scales efficiently, improving with more transactions.
Decentralization	Highly decentralized via node consensus.	Decentralization can vary, potentially decreasing if network participation is limited.
Transaction Cost	Higher fees due to mining expenses.	Minimal to no fees as mining isn't required.
Energy Consumption	High, especially in PoW systems.	Lower, as energy-intensive mining is absent.
Applications	Used in cryptocurrencies, smart contracts, and supply chains.	Suitable for fast transactions, IoT, micropayments, and real-time data.

The comparative analysis presented in Table 2.4 elucidates the fundamental distinctions between Blockchain and DAG technologies, highlighting their respective structural designs, consensus mechanisms, and operational efficiencies. In contrast, blockchain systems rely on a linear series of blocks, where each block contains a group of transactions. The addition of new blocks to the chain is governed by agreement mechanisms like PoW or PoS. These protocols preserve the security and trustworthiness of the blockchain but often come with scalability and efficiency challenges. This linear architecture inherently limits transaction throughput, as each block must be sequentially processed, leading to potential bottlenecks and scalability challenges. In contrast, DAG structures transactions in a non-linear, acyclic graph where each transaction can confirm multiple previous ones, facilitating parallel validation processes. The DAG structure offers an advantage in scalability and transaction speed by allowing multiple transactions

to be processed concurrently. This parallel processing capability eliminates the limitations imposed by block intervals, enabling the system to manage a growing volume of transactions efficiently. Moreover, the absence of traditional mining in DAG systems results in lower energy consumption and reduced transaction fees, making them more efficient for certain applications. However, the degree of decentralization in DAG networks can vary and may be influenced by the level of adoption and network participation, potentially impacting security and resilience. Understanding these technical differences is crucial for selecting the appropriate DLT tailored to specific use cases, performance requirements, and scalability needs.

However, DAG-based totally DLTs additionally face particular protection challenges, consisting of the risk of double-spending in low-pastime intervals. notwithstanding those worries, advancements in DAG consensus algorithms, which include modifications to the Tangle's attachment mechanism in IOTA, display promise in retaining transaction balance and resisting certain assault vectors, making DAGs a feasible solution for high-scalability packages that blockchains conflict to support successfully (Ferraro et al., 2020).

2.2.4 DAG architecture in networking

The DAG structure within DLT presents a novel approach to transaction validation by moving away from traditional block-based confirmation methods. Unlike blockchains that aggregate multiple transactions into blocks and validate them sequentially, DAG-based systems validate each transaction individually, creating a continuous, block-free network of transactions. This architecture allows for high concurrency, as transactions can validate each other asynchronously, significantly reducing latency and increasing throughput. The inherent parallelism of DAG structures allows for multiple transactions to be processed simultaneously, which is particularly beneficial in high-demand network environments like IoT and 5G networks (Peng et al., 2023).

In a DAG-based DLT, each transaction node typically references and validates previous transactions, forming a directed link to earlier transactions without requiring a formalized

block. This method not only accelerates the validation process but also improves network expandability by removing the requirement for global consensus on blocks. For instance, IOTA's Tangle structure enables each new transaction to confirm two previous ones, allowing transactions to progress without waiting for block formation, which is a core limitation in conventional blockchain models (Chen et al., 2022).

The decentralized nature of transaction validation in DAGs enhances the resilience of the network against delays and bottlenecks that can occur in block-based systems. Without the need for miners or block creation, DAG-based ledgers can accommodate a significantly higher transaction volume and avoid single points of delay that traditional blockchains face. This decentralized validation method also mitigates the risks associated with centralized nodes or mining pools, as seen in blockchains. By utilizing individual transactions as confirmation points, DAG systems can maintain consistency and security in the ledger with a far lower resource expenditure (Wang et al., 2022).

2.2.5 DAG and Network Performance

DAG-based DLTis highly beneficial in high-throughput networks due to its ability to address scalability and speed limitations found in traditional blockchain systems. Unlike the sequential block formation in blockchain, DAG organizes transactions in a multi-node structure that allows concurrent processing of multiple transactions. This structure significantly improves transaction throughput by enabling parallel validation, which is ideal for networks with high data demands, such as IoT and 5G systems (Chen et al., 2022).

One of the primary advantages of DAG in high-throughput networks is its scalability. DAG's non-linear design removes the need for block-based confirmations, reducing latency and improving overall transaction speed. For instance, the TEEDAG architecture has demonstrated significantly higher throughput compared to traditional blockchain systems by leveraging parallel chains, allowing numerous transactions to be processed and validated simultaneously. This model is especially effective in high-demand

networks, as it accommodates a far greater transaction volume without compromising performance (Lu & Jiang, 2023).

DAG also enhances network efficiency by supporting high concurrency in transaction processing. Models such as the Community-Assisted DAG (C-DAG) have shown that dividing network nodes into smaller communities and enabling parallel transaction processing within these communities can substantially increase throughput. The C-DAG model utilizes event-driven algorithms to ensure final consistency across the network, making it highly efficient and resilient in large-scale environments where rapid transaction speeds are critical (Zhang et al., 2020).

In summary, DAG's inherent scalability and transaction speed make it an optimal choice for high-throughput networks, as it supports the concurrent processing of transactions without the bottlenecks typical of blockchains.

2.2.6 Use cases of DAG in SDNs

DAG technology offers substantial benefits for SDN environments, especially in enhancing decentralization and supporting QoS integration. In large-scale SDNs, DAG-based systems facilitate decentralized control and improve scalability by distributing network control responsibilities across multiple nodes, reducing dependency on centralized controllers. The SliceBlock framework, for example, utilizes DAG-based blockchain to achieve secure and efficient network slicing in 6G environments. This approach replaces traditional consensus mechanisms with Proof of Space, which reduces resource consumption and supports QoS-aware slicing that prioritizes network slices based on QoS demands, enabling optimal data transmission and load balancing across SDN controllers (Abdulqadder & Zhou, 2022).

DAG’s structure is also advantageous in multi-domain SDN environments for QoS management. By enabling hierarchical controller arrangements, DAG allows inter-domain flow decision-making based on QoS metrics such as effective delay and

bandwidth requirements. This decentralized approach is instrumental in delivering E2E services with consistent QoS across interconnected SDN controllers, a critical need in networks supporting real-time applications like augmented reality and streaming. The hierarchical arrangement minimizes latency and enhances bandwidth utilization, ensuring that QoS demands are met across diverse domains (Lee et al., 2022).

Additionally, DAG-based models in SDNs allow for more efficient topology discovery and routing in wireless mesh networks. By decentralizing the control Layer, DAGs reduce the load on SDN controllers and allow for faster local routing adjustments without requiring centralized intervention. This decentralized QoS-aware topology discovery, which leverages DAGs, enables networks to dynamically adjust routing paths based on QoS requirements and resource availability, minimizing network traffic overhead and improving reliability and response times (Elzain & Yang, 2018).

2.3 RL

RL is a branch of machine learning where an agent learns the best strategy by interacting with its environment and maximizing the cumulative rewards over time. The process is typically modeled as a Markov Decision Process (MDP), characterized by states (S), actions (A), transition probabilities P(s′|s,a), and a reward function R(s,a). RL has shown significant success in optimizing real-time traffic management within SDNs, where it dynamically adjusts to changing network conditions.

The fundamental objective of RL is to determine a policy π(a|s) that maximizes the anticipated long-term reward Gt over time, given by:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The discount factor γ is critical in RL, balancing the significance of prompt versus long-term rewards. Through exploration and exploitation, the agent iteratively improves its policy, ensuring better performance over time. In the Q-learning algorithm, a model-free

RL approach, the action-value function Q(s,a) is used to evaluate the anticipated long-term reward, guiding the agent's decisions:

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)]}$$

Here, α is the learning rate, and *Q(s,a)* is updated iteratively to converge toward the optimal policy.

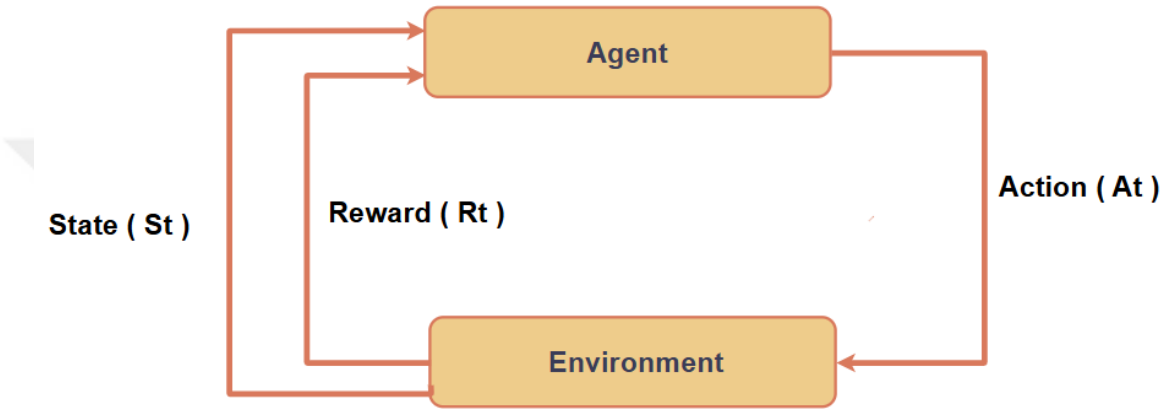


Figure 2.7 RL process in SDNs

Figure 2.7 illustrates the RL loop. The agent observes the current state St, performs an action At, and receives a reward Rt based on the environment's response. This feedback loop allows the agent to refine its policy iteratively, enhancing decision-making and achieving optimal performance.

RL in SDNs allows for dynamic traffic management by learning from real-time network feedback, optimizing parameters such as latency, bandwidth, and throughput. Specifically, RL models like Q-learning have shown superiority over static routing protocols like OSPF by adapting to varying traffic demands, minimizing congestion, and maximizing QoS.

Moreover, RL has been integrated into DAG-based DLTs to enhance security and scalability. For instance, RL has been employed in DAG-based systems like IOTA to identify and mitigate threats, such as DDoS attacks. By analyzing node resource usage

patterns, RL models enable real-time anomaly detection, strengthening the integrity of decentralized systems.

For large-scale SDN environments, multi-agent RL approaches can be implemented, where each agent operates within a localized network segment. These agents collaborate to optimize their routing policies collectively, ensuring global QoS adherence. Multi-agent systems address scalability challenges in single-agent models, enhancing adaptability and efficiency in complex, distributed networks.

2.3.1 RL fundamentals

RL operates through interactions between key elements: the agent, state, action, and reward. In this learning paradigm, the agent is the decision-maker within the RL framework, tasked with exploring the environment to achieve specific goals. The agent must decide its decisions influenced by the environmental state, which encapsulates all relevant information about the current situation. States serve as snapshots of the environment, providing the agent with context for making informed decisions (Lin et al., 2020).

The action refers to any decision or move the agent makes in response to a given state. In the case of SDNs, actions might include modifying routing paths or adjusting traffic flow to manage network loads effectively. These actions influence the transition of the environment from one state to another, following a MDP framework, the probability of transitioning to the next state in a RL system depends solely on the current state and the action taken, adhering to the Markov property (Zhao et al., 2019).

A reward is the feedback provided to the agent after executing an action, serving as a measure of the action's success or failure. The fundamental goal of RL is to maximize the cumulative reward, which helps the agent progressively refine its behavior towards optimal decision-making. In multi-agent network systems, for example, this reward structure may reflect both individual and collective benefits, making it crucial for agents

to adopt actions that optimize both local and global objectives, particularly when balancing load and minimizing latency in network traffic (Wu et al., 2020).

2.3.2 Types of RL algorithms

RL is a study area that contains several types of algorithms in order to optimize the decision-making process of an agent. Among them, some popular ones involve Q-learning, SARSA, and Deep Reinforcement Learning (DRL) because of their distinct methodology and application areas.

Q-learning is an off-policy algorithm in that the agent learns a completely separate policy from the actions it actually executes. Q-learning works based on updates to a utility table, called the Q-value table, regarding the utility of taking actions at particular states. The key update rule from the Bellman equation dictates an optimal policy where the agent explores actions to capture future rewards. The general advantage of Q-learning is that it identifies optimal actions without necessarily following a strict policy; hence, it works in situations where decisions must not be rigid (Sewak, 2019).

At the same time, SARSA is an on-policy algorithm, meaning that the agent learns through the policy it most actually follows. The update rule for SARSA depends on the next state-action pair, given consideration to the immediate action chosen by the agent. It is thus more consistent with the current policy, hence more conservative than Q-learning, as it risks being shy in choosing risky actions due to the response by the policy to environmental dynamics. SARSA is particularly useful in stable environments where adherence to a policy is critical for avoiding adverse outcomes (Wang et al., 2013).

DRL incorporates neural networks in order to approximate the Q-values and hence enables the algorithm to scale to any high-dimensional state space. This approach, epitomized by techniques like Deep Q-Networks, merges deep learning with RL, which would enable the agent to learn directly from raw inputs, such as images or complex simulations. Variants, including Double DQN and Deep SARSA, avoid overestimation

in value functions and allow for extensive experience replay, hence enhancing training stability and efficiency in complex environments (Xu et al., 2018).

2.3.3 RL applications in networking

RL can be highly applied in this field for effective resource management by adapting to routing and path optimization in networking. The RL routing approaches enable networks to adapt their paths in real time, honored by the current load of traffic, which reduces latency and optimizes path selection based on complex metrics such as QoS and network congestion.

The fixed algorithms of shortest-path routing in traditional routing usually lead to congestion because they cannot adapt in real time. Adaptive routing based on RL allows the network to dynamically select paths by evaluating the condition of traffic flows and possible delays. For example, route selection based on real-time traffic states avoids congestions in the predictive RL approach of adaptive routing. As a result, packet delivery times are faster, while the network resiliency to bursts of traffic flow is improved compared to other techniques (Desai & Patil, 2017).

RL is applied to QoS-based routing. In such frameworks, adaptive algorithms using neural networks estimate traffic and optimize routing paths with respect to QoS requirements—like minimum average packet delivery time or selecting the low-latency route. These systems can adapt routing decisions with environmental states of queue sizes at routers and thus outperform other classical algorithms like shortest-path routing in high-traffic or complex network environments (Mellouk et al., 2007).

DRL thereby enriches these applications by utilizing deep neural networks for handling large-scale input data, allowing the efficient routing even on large and complex network topologies. DRL models, like those applied in SDN, are capable of adapting to the time-varying traffic load through runtime view-based route optimization using real-time metrics such as link delay and packet loss. It is evident that this model outperforms other

Dijkstra-based traditional algorithms by reducing latency and packet loss, hence providing better accommodation for QoS requirements (Casas-Velasco et al., 2021).

2.3.4 Challenges in applying RL to SDNs

Applications of RL in SDNs are quite challenging; the high computational load and the real-time learning process make it difficult. Generally, the integration of RL into SDNs requires intensive computational resources due to the way it is continuously observing network conditions to adapt. All this is further exacerbated by the use of deep RL models for processing high-dimensional data in SDNs, which at all times encompasses complex neural networks that further increase computational overhead, hence defeating real-time processing (Asseman et al., 2021).

The much larger state-action space in network environments is also a significant challenge. In dynamic network environments with many possible routing paths and traffic states, processing and evaluating vast numbers of state-action pairs would have to take place through the RL algorithms, which may become computationally prohibitive. Researchers have found methods for reducing the dimensionality of state abstraction and function approximation, but even in these cases, real-time applications may not be achieved as the networks are fast in velocity and changeable (Kelly & Heywood, 2018).

Other challenges include keeping the decision-making latency low. An SDN has to make routing decisions in milliseconds; otherwise, it delays a network application and degrades QoS. Classic RL models cannot meet this real-time requirement since, after training, it takes too long to explore by the agent and achieve the optimal policy.

It has been proposed that hybrid approaches, like combining RL with evolutionary algorithms or distributed architectures, could speed up learning by reducing dependency on sequential data processing, but these also need optimization in order to be feasible for real-time applications (Younus et al., 2021). In other words, the application of RL to SDNs faces a double challenge: high computational cost versus real-time demand for

maintaining QoS, which requires further development of efficient low-latency RL methods.

2.4 QoS in Networking

QoS encompasses a range of methods and protocols designed to ensure that key network performance metrics, including bandwidth, latency, jitter, and packet loss, meet the specific requirements of various applications. QoS is especially critical for latency-sensitive and high-priority applications like VoIP, video streaming, and online gaming, where poor performance can severely degrade user experience (Gandhi & Sajnani, 2020).

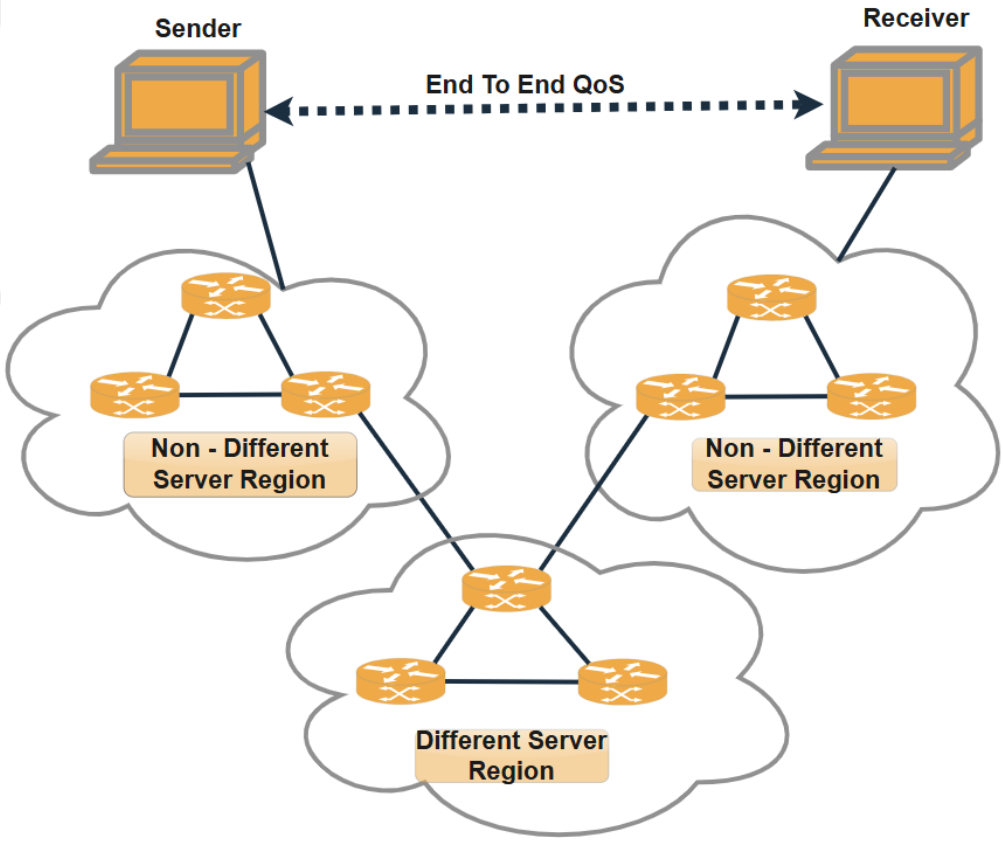


Figure 2.8 E2E QoS in networking across server regions

Figure 2.8 illustrates the data transmission flow between a sender and a receiver while traversing different server regions. It highlights the distinction between "Non-Different Server Regions," where routers within the same network or server region handle the data,

and "Different Server Regions," where data travels between distinct server zones. The E2E QoS requirements, such as bandwidth, latency, and jitter, are maintained throughout the transmission path to ensure optimal network performance. By showcasing how data flows are managed across different regions, this figure emphasizes the importance of prioritizing traffic and maintaining QoS standards in complex networking environments. It demonstrates the capability of advanced SDN frameworks to dynamically adapt to diverse traffic conditions and optimize resource allocation across various network segments.

A key aspect of QoS is traffic prioritization, where network traffic is classified into categories and assigned varying levels of importance. For instance, real-time applications like voice and video data are prioritized over less time-sensitive activities, such as file downloads. Approaches like Differentiated Services (DiffServ) and Integrated Services (IntServ) are commonly used to allocate resources effectively, ensuring that the QoS requirements of each application are met (Chien & Kim, 1997).

In the context of SDNs, QoS management becomes more dynamic and efficient due to SDN's centralized control and programmability. SDNs enable real-time traffic flow adjustments based on changing network conditions, optimizing resource allocation and ensuring QoS compliance in diverse and complex networking environments (Karakus & Durresi, 2017).

Wireless networks also rely heavily on QoS to handle unique challenges such as node mobility and shared channel access. Techniques like cross-layer design, resource reservation, and adaptive scheduling are utilized to ensure adequate performance for both voice and data traffic under dynamic and unpredictable conditions (Chen & Cobb, 2006).

2.4.1 Definition of QoS in networking

QoS, in networking, is a set of several core metrics that define the capability of the network to meet certain application requirements. Bandwidth describes the maximum

possible data transfer rate in any one network connection and decides how much information can really be pumped across a network during any given time. Adequate bandwidth provisioning is crucial for services such as video streaming and online gaming, which require high data throughput to maintain smooth and uninterrupted performance. Ensuring these needs are met is essential for delivering high-quality user experiences. In reliable bandwidth management, prioritization of vital traffic is facilitated by the availability of a steady flow of data even in conditions of network congestion (Kota, Pahlavan, & Leppanen, 2004).

Latency is the time difference in the exchange of messages, which is crucial in the case of real-time applications related to VoIP and interactive gaming. Lower latency means better user experience because data transfer time is minimized, and hence there can be more spirited communications. Network designers use various means of reducing latency, which is the time packets take through a network from source to destination. Means employed in modern networks include caching and routing optimization (Gandhi & Sajnani, 2020).

Reliability can be defined as the dependability of a network in terms of continuity of services and accuracy of data transmission. It guarantees that the services the network provides will maintain standards over time despite issues like packet losses and network failures. In that regard, such networks are mostly designed so they can incorporate a degree of redundancy, error correction protocols, and resource allocation methods that ensure consistent connectivity with data integrity under various circumstances (Allison, 1998).

2.4.2 QoS metrics and parameters

Accordingly, QoS metrics are serious issues that will define how well the network is capable of handling data flows in accordance with its continuity and effectiveness. Such parameters as FET, MEE, APL, ANoL and ANL are of prime importance for defining network performance in the class of high-demand applications including real-time data transfer and multimedia streaming.

FET measures the duration required to establish a connection between nodes within the network. Such a metric is crucial in latency-sensitive applications, whereby any setup delay will result in degraded service quality. For instance, in an MPLS network, the setup time of a flow should be negligible to facilitate efficient and predictable service quality. Evidence of this—that faster setup times directly lead to a better user experience since waiting times for initial data exchanges are reduced—is provided by Niu et al. (2003).

The message exchange process pertains to the reliability and efficiency of data packets being transferred between endpoints. In real-time applications, the rate of message exchanges needs to be optimized so that the flow of data is smooth, continuous, and sustained for QoS in packet-switched networks. Quite often, queuing models and different methods of flow control are employed in high-performance systems to ensure that message exchanges do not get clogged and help in achieving low latency and minimal packet loss. These methods have ensured that the congestion of networks has been controlled well, which has enhanced general QoS (Hua & Qu, 2003).

2.4.3 Traditional QoS protocols

Traditional QoS protocols (integrating OSPF, Dijkstra algorithm) are fundamental in providing a high quality and dependable routing method in the network infrastructure. OSPF is a link-state routing protocol, which uses Dijkstra's algorithm and the Shortest Path First (SPF) method to calculate optimal paths between nodes located in different networks. The protocol offers dynamic network topologies and scalable operation, allowing routers to exchange information concerning the way conditions on networks are changing so that it can quickly compute a route for congested or disrupted links (Chun 2003).

When using the Dijkstra algorithm, start always with a lowest cumulative link cost first strategy, this ensures getting closer to capacity preservation of network components. OSPF, however, can be extended with Field 3 so as to sustain the requirements of QoS by introducing qos metrics like delay and bandwidth hence improving its ability in fulfilling a certain state request. Some enhanced versions of OSPF, where the path

calculation takes into account queuing delay and link capacity have been shown to provide better load distribution with reduced E2E delays in an IP network environment useful for QoS-centric applications (Ashwini et al., 2011).

Moreover, QoS extensions to OSPF uses Dijkstra's algorithm for path selection based on specific QoS constraints. Consequently, the network traffic is balanced and congestion reduced [12]. These adaptations allow the protocol to find not only a minimax path, but also an optimal route under some conditions including high-priority data flows that optimize both reliability and QoS (Apostolopoulos et al., 1999).

2.4.4 The role of QoS in SDN

QoS is important in SDNs for the management that demands dynamic network environments. SDNs can realize the control of network resources centrally in real time and can handle QoS management efficiently by dynamically adjusting bandwidth, latency, and prioritization based on network state. This is important in high-performance applications whose network conditions are so transient, and assures strict QoS guarantees in cases such as multimedia streaming, cloud services for seamless data delivery (Karakus & Durresi, 2017).

QoS is chiefly implemented through resource reservation and prioritization of traffic—an administrative role managed by the SDN controller. With a global view of the network, the controller can dynamically perform resource allocation, traffic rerouting, and data flow prioritization in real time to achieve optimized network performance. This leads to not only fine-grained control over the patterns of traffic but also drastically reduces latency while ensuring that key services are not affected by network congestion (Xu et al., 2015).

Second, the dynamic flow management in SDN can provide differentiated handling of traffic types through mechanisms such as DiffServ, which can reactively change queue assignments and flow priorities of the network. Indeed, this capability is very useful in

the multi-media-based SDNs because the quality metrics that vary—losing packets and delays—directly affect the Quality of Experience provided to the users. Recent research works have experimentally presented the QoS mechanisms based on SDN, which can achieve up to 47% gains regarding data transfer times, revealing the potential of SDN for QoS enhancements in dynamic environments (Oliveira et al., 2018).

3. LITERATURE REVIEW

The literature review seeks to offer a thorough understanding of the fundamental concepts and previous studies related to SDN, DLT, and RL. This section examines the current state-of-the-art solutions, highlighting their strengths and limitations, and sets the foundation for identifying the research gap that this thesis addresses. Specifically, the review delves into the integration of SDN with DLT, including Blockchain and DAG-based systems, and explores RL-based frameworks for adaptive and scalable traffic management.

3.1 Integration of SDN and DLT in Networking

The integration of SDN and DLT has been widely studied to enhance network security, decentralization, and scalability. Blockchain-based solutions, such as QoSChain (Karakus & Guler, 2021), provide robust transaction validation and distributed traffic management. However, due to the sequential nature of block processing, these solutions face scalability and latency limitations. To overcome these challenges, Blockchain-assisted source routing mechanisms have been proposed, demonstrating improvements in traffic management and network efficiency (Karakus, 2023). These approaches leverage Blockchain's security and immutability to improve inter-AS routing, but they still suffer from transaction delays and high computational overhead, making them less efficient for real-time traffic management.

Similarly, Blockchain-based traffic management frameworks have been explored to optimize cross-ISP and inter-AS routing coordination. RoutingChain (Karakus & Guler, 2020) introduced a proof-of-concept Blockchain-enabled inter-AS routing framework, designed to eliminate centralized mediators, enhance security, and reduce QoS signaling overhead. Experimental results demonstrated that Blockchain can be effectively applied to E2E QoS-based routing, reducing security and privacy concerns in inter-AS communication. Additionally, Smart Contract Chain (SC2) proposed by Karakus et al. (2023) extends these principles by integrating smart contracts for intelligent cross-network QoS-based service provisioning in Next-Generation Networks (NGNs). SC2

enhances network visibility, control, and operational efficiency, mitigating controller overhead in SDN environments. These Blockchain-based solutions introduce greater decentralization and transparency, yet still struggle with high processing latency and energy consumption due to consensus mechanisms.

Alternatively, DAG-based DLT systems offer faster and more efficient transaction processing, making them more suitable for high-throughput network environments. Unlike traditional Blockchain architectures, DAG structures support parallel processing, reducing transaction confirmation times and increasing overall network performance. However, most existing studies primarily focus on security and transaction validation rather than comprehensive QoS management in SDN environments. This highlights the need for further research on integrating DAG-based DLT with SDN traffic management to ensure not only decentralization and security but also real-time adaptive routing and QoS optimization.

Future research should focus on developing solutions that address both security and decentralization while ensuring efficient real-time routing and QoS enhancement in SDN and SDON environments.

3.2 DAG-Based Distributed Ledger Technologies

Xu et al. (2023) analyze the performance of Phantom, an alternative DAG model, under varying network loads. Their findings indicate that DAG's structure allows for simultaneous validation, mitigating congestion in high-volume scenarios. However, they note that consensus algorithms remain a bottleneck, particularly under adversarial conditions where malicious actors can exploit asynchronous models.

Miller et al. (2022) investigate the applications of DAGs in broader distributed systems beyond IoT, focusing on smart contracts and micro-transactions. The researchers point out the reduced latency inherent in DAGs, which is pivotal for time-sensitive operations. Yet, scalability trade-offs still pose constraints, requiring advancements in consensus

mechanisms to address vulnerabilities related to transaction ordering and validation efficiency.

Seongjoon Park and Hwangnam Kim (2019) conducted a study proposing a DAG-based DLTnamed PowerGraph for managing electricity transactions within Smart Grids. Their study emphasizes the drawbacks of conventional blockchain systems, including inefficiencies and delays caused by the resource-heavy consensus methods like Proof-of-Work. PowerGraph addresses these issues by eliminating the need for block generation, allowing each transaction to be validated individually, thereby improving transaction confirmation speed by over five times. This approach enhances the scalability and efficiency of Smart Grids.

Watanabe and Shoji (2023) presented a study focused on using DAG-based DLTs for content management in Beyond 5G (B5G) networks. Their framework, which includes a partially ordered DAG-based ledger (DAG-DL), showed that it could handle high censorship resistance better than conventional blockchains even in complex network conditions involving multiple pathways and multi-hop relays. The simulations demonstrated that DAG-DL could efficiently manage transaction recording with reduced delays, ensuring high network performance.

Cullen et al. (2020) investigated the robustness of DAG-based ledgers, particularly the IOTA system, in Internet of Things (IoT) environments. Their research addressed the security risks associated with the "parasite chain attack," which compromises the integrity and permanence of the ledger. By employing a Markov chain model, they examined vulnerabilities and proposed an improved tip selection algorithm to strengthen ledger security. This work contributed valuable insights into enhancing the resilience of DAG-based DLTs against targeted threats.

Caixiang Fan (2019) conducted an extensive performance analysis of DAG-based DLTs for IoT, focusing on the IOTA protocol. Fan’s research included deploying a scalable infrastructure involving 40 smart home nodes and evaluated the system’s transaction speed, scalability, and effectiveness in micropayment contexts. The study found that

DAG-based ledgers could break through the "blockchain trilemma," which suggests it is challenging to achieve decentralization, scalability, and security simultaneously. The findings demonstrated that DAG-based DLTs are effective in supporting IoT infrastructures, maintaining high transaction throughput and low latency.

3.3 RL in Traffic Management

RL has been utilized to enhance routing and traffic management in SDNs by enabling adaptive decision-making based on current network conditions.

In the literature, various studies have employed RL to address traffic management challenges. Many of these approaches involve learning specific parameters prior to applying conventional routing algorith studies like Chen et al. (2020) and Hossain (2020) have demonstrated the use of Q-learning for adaptive path selection, improving QoS metrics such as latency, bandwidth utilization, and throughput.

Yu et al. (2018) introduced Deep Deterministic Policy Gradient Routing Optimization Mechanism (DROM) for SDN, aimed at delivering a universal and customizable routing optimization. DROM enhances network operation and maintenance through a continuous-time black-box optimization technique that improves key metrics such as delay and throughput. Sun et al. (2019) developed Time-Relevant DRL For Routing Optimization (TIDE), an intelligent network control architecture designed to automate routing in SDN. TIDE utilizes a complete “collections-decision-adjustment” loop to achieve intelligent routing control in a data-transmitting network.

Stampa et al. (2017) created a DRL agent that optimizes routing by automatically adapting to live traffic conditions and providing tailored settings intended to reduce network latency. Pham et al. (2018**)** applied a DRL agent equipped with convolutional neural networks under the Knowledge Defined Networking (KDN) paradigm to enhance QoS-aware routing. Guo et al. (2020) proposed DQSP, a DRL-based QoS-aware secure

routing protocol. While ensuring QoS, DQSP extracts insights from past traffic demands by interacting with the network environment and dynamically refining routing policies.

Rischke et al. (2020) devised a classic tabular RL framework QR-SDN that encodes individual flow routes directly into its state-action space. QR-SDN is the first RL-based SDN routing solution to allow multiple routing paths between a specific source (ingress) switch and a destination (egress) switch while maintaining flow integrity.

Chen et al. (2020), an RL-based method for SDN Traffic Engineering (TE) is introduced, aiming to maximize throughput and minimize delay without relying on exact mathematical formulations. This method represents network conditions through extensive data, allowing more adaptable routing decisions. The reward function is designed to either encourage increased throughput or reduce it, depending on network needs. However, message exchange and synchronization in large-scale, multi-controller SDN environments remain unresolved. Although BGP can be utilized for inter-domain synchronization, it is both intricate and problematic in SDN contexts. Meanwhile, although OpenFlow enables cross-domain communication, a lack of standardization has hindered its widespread adoption.

Ye et al. (2024) proposes a multi-agent DRL method called MDRL-TP, which takes advantage of SDN's multi-threaded measurement capabilities and socket-based messaging to enhance synchronization speed, reliability, and real-time route optimization. Recent progress in AI has greatly impacted computer science, and SDN's centralized nature supports AI-driven resource management. However, current SDN routing algorithms still face problems such as underutilized link capacity and inadequate responsiveness to real-time variations.

To tackle these concerns, Chen et al. (2022) introduce a RL-Based Multipath Routing (RLMR) framework, leveraging MDP and Q-Learning to deliver adaptable routing. Additionally, they incorporate Forward Efficiency (FE) to measure how effectively multipath routing uses link bandwidth. By maintaining a centralized, software-centric structure, SDN has significantly enhanced network service quality and adaptability,

simplifying policy configurations, traffic engineering, and monitoring. Nevertheless, with diverse applications requiring conditions like minimal latency and low packet loss, both QoS policies and link utilization become paramount.

Chiu et al. (2022) present RED-STAR, an RL-driven system that selects optimal routes by combining network state data and service type. This solution incorporates dedicated protocols and a deep learning-based traffic classification module to manage services more effectively. The widest-path problem, focused on finding routes with the greatest transmission capacity, remains crucial in a variety of industries. Ke et al. (2023) propose QWPRA, a Q-learning solution for widest-path routing in SDN. This approach calculates a reward based on link bandwidth to locate the path offering the highest bandwidth and has proven adept at adapting to varying network states, evaluating bandwidth, loss rates, and traffic to decide the best path.

In Kim et al. (2022), a DRL-based routing system relies on the DDPG algorithm within SDN to control traffic routing. An agent learns link weights via an Aggregated Traffic Volume Matrix (ATVM), aiming to reduce both delays and packet losses. The SDN controller then applies these weights when configuring routing paths. An M/M/1/K queue model is utilized for offline training, thus preventing service quality declines during the DRL learning phase.

Sarwar et al. (2023) explore the application of Multi-Agent RL (MARL) for traffic flow management, focusing on how it improves autonomous vehicle routing on road networks. Their study tests techniques like Multi-Agent Advantage Actor-Critic (MA2C) in simulated traffic scenarios using SUMO. Results indicate that MARL strategies, particularly MA2C, optimize traffic flow and outperform traditional methods by handling complex, dynamic traffic conditions effectively.

J. Lee, J. Chung, and K. Sohn (2020) examined RL for adaptive traffic signal control across multiple intersections. Their study employed a Deep Q-Network (DQN) model which utilized traffic images to optimize signal timings by overcoming the complications of dealing with expansive state-action spaces. This RL model aimed to improve traffic

flow coordination and outperformed traditional fixed-signal operations and partially coordinated multi-agent RL approaches. The use of Convolutional Neural Networks (CNNs) to approximate the Q-function provided the flexibility needed for real-time traffic signal adjustments, resulting in enhanced overall traffic management performance.

Wang et al. (2022) investigate DRL applications for real-time traffic signal management in urban settings. Their research highlights the advantages of DRL for adaptive traffic light control, which reduces congestion and improves response times compared to conventional fixed-schedule systems. Simulations conducted using real-world traffic data demonstrate significant reductions in vehicle wait times and fuel consumption.

In another significant work, Azevedo et al. (2021) apply Q-learning and DQN for traffic signal optimization. Their findings reveal that these methods adaptively learn to balance traffic flow by considering historical data and real-time sensor inputs, resulting in more effective signal adjustments and reduced congestion.

K. Yau and co-authors (2017) conducted a comprehensive review of RL models applied to Traffic Signal Control (TSC), focusing on how RL agents select optimal actions to manage traffic phases and timing. This survey analyzed state representations, action spaces, reward functions, and the complexity of various RL implementations, providing a solid foundation for future research by identifying existing gaps and potential enhancements in RL-based traffic management systems.

Coskun, Baggag, and Chawla (2018) explored DRL for optimizing traffic light operations. Their work introduced a novel reward function that factored in both traffic flow and delay, employing deep Q-learning and policy gradient methods to minimize vehicle travel time. Simulation results confirmed the effectiveness of these RL-based strategies in achieving efficient traffic signal management, especially under varying traffic conditions.

Namrata S. Jadhao and A. S. Jadhao (2014) demonstrated the application of the Q-learning algorithm for managing traffic signals at crossroad points. By using statistical traffic data, the study optimized signal timings to improve the overall flow of traffic under both under-saturation and over-saturation conditions. Their results validated the Q-learning algorithm's ability to adapt traffic signals effectively to real-time changes in traffic volume.

3.4 Synergy of SDN, DAG-based DLT, and RL in Traffic Management

The convergence of RL, SDN, and DLTconstitutes an innovative strategy for addressing consensus and resource optimization within distributed environments.

Through advanced RL approaches such as DRL networks can flexibly allocate resources in real time under SDN control. Meanwhile, DLT supports synchronization across multiple SDN controllers, ensuring a globally consistent view of the network and maintaining data integrity as well as transparency. This integrated approach has the potential to bolster scalability, efficiency, and security in a variety of distributed infrastructures, including the Industrial Internet of Things (IIoT).

In particular, Software-Defined IIoT (SDIIoT) has evolved from merging SDN with IIoT, enhancing overall system administration. Multi-SDN designs, as noted by Luo et al. (2020b), enable distributed control planes for large-scale data processing; however, achieving controller consensus in such contexts remains complex. To address this, Luo et al. (2020a) introduced a DLT-driven SDIIoT that achieves global agreement on network states, boosts energy efficiency with MEC resources, and employs DRL in a POMDP framework for streamlined DLT deployment. While this approach strengthens SDN’s security, it also brings new challenges, especially when filtering or updating malicious traffic flows. In such scenarios, deep learning emerges as a critical element for safeguarding the SDN-DLT ecosystem.

The Energy Internet (EI) similarly benefits from DLT adoption, albeit it encounters hurdles like credit risks and utility optimization in its distributed energy resource settings. In response, Cao et al. (2022) proposed a Blockchain-Assisted Software-Defined Energy Internet (BSDEI) that ensures reliable transactions via smart contracts. They model the trading workflow as a three-layer Stackelberg game, employing a distributed policy gradient algorithm to handle incomplete information. The resulting DLT-based energy trading platform is then integrated into a middleware solution.

Additionally, SDN has been woven into the Internet of Vehicles (IoV) to enhance applications such as autonomous driving and traffic monitoring, leveraging spatial crowdsourcing to efficiently handle data collection. Given the substantial data volume in IoV systems, privacy and security pose significant issues. To tackle these, Lin et al. (2021) proposed the DRL and Blockchain-based Spatial Crowdsourcing System (DB-SCS), employing a multi-DLT structure and hierarchical task coordination. This design allows for dynamic selection of block size, consensus algorithms, and generation policies to safeguard privacy while optimizing performance. Since IoV is inherently decentralized connecting vehicles and VANETs its large-scale data management is further complicated by diverse QoS requirements. Fog computing addresses some of these constraints by offloading processes to nearby nodes, yet as IoV systems expand, maintaining scalability, efficiency, and security becomes paramount. In that vein, Alotaibi and Alazzawi (2021) proposed SaFIoV, an RL-based mechanism that balances loads and secures communications in SDN-powered fog-based IoV by allocating tasks across various layers.

Hossain (2020) explores the application of RL for QoS management in SDNs, demonstrating how predictive models can proactively modify network paths to avert congestion. However, the combination of RL and DLTs—especially where DAGs facilitate decentralized validation—remains relatively under-investigated. The synergy of RL's adaptive learning capabilities and a DAG's streamlined validation mechanism appears promising, yet further research is necessary to realize its full potential.

Zhang and colleagues (2020) examined how RL techniques could be integrated into SDN routing to optimize traffic management. They developed a Q-learning-based model that adaptively responded to real-time network updates for refined path selection. Their results indicated that RL methods outperformed traditional algorithms—such as Dijkstra's—in dynamically adjusting to fluctuating traffic conditions and preserving QoS benchmarks. Tests conducted in an SDN emulation setup using Mininet consistently showed reduced latency and improved bandwidth allocation when compared with conventional methods.

Chen (2020) embedded RL within SDNs to address both load balancing and traffic routing, adopting a multi-objective optimization perspective. This work highlighted the benefits of model-free RL approaches, such as Q-learning, for autonomously adapting routing choices to changes in the network state. Findings suggested that RL effectively managed QoS by continually reacting to traffic variations, thus preventing congestion and maximizing resource utilization. Nonetheless, the study also noted that training RL models can become computationally intensive in large-scale SDN environments.

H. Ghalwash and C. Huang (2020) introduced a congestion control methodology for SDN environments that harnessed RL feedback to determine routing based on QoS parameters like latency and bandwidth. Their empirical results demonstrated notable improvements in the network's responsiveness to congestion, surpassing more classical protocols like OSPF in terms of overall QoS outcomes. By merging SDN with machine learning-oriented simulations, the study emphasized RL's adaptability while also noting the substantial hurdles of real-time model convergence. To our knowledge, the methodology presented here in DQRL is the first to integrate SDN with a DAG-based DLT for a novel network management paradigm driven by QoS metrics in path selection, while also leveraging RL through AS networks.

Table 3.1 Comparative analysis of studies integrating SDN, DLT, and RL in traffic management

Studies	SDN	DLT	RL	RL Objective	RL Mission	Functional Area
Ke et al. (2023)	✓	×	✓	Data rate, loss	Path Finding	Networks
Chen et al. (2022)	✓	×	✓	Link Bandwidth Utilization	Path Finding	Networks
Chiu et al. (2022)	✓	×	✓	QoS	Path Finding	Networks
Kim et al. (2022)	✓	×	✓	Min Delay & Loss	Load-balancing	Networks
Chen et al. (2020)	✓	×	✓	Data Rate & Delay	Path Finding	Networks
Karakus & Guler (2020)	✓	BC	×	Bandwith & Delay	Path Finding	Networks
Karakus et al. (2021)	✓	BC	×	Bandwith & Delay	Path Finding	Networks
Alotaibi and Alazzawi (2021)	✓	BC	✓	Utility & Delay	Load-balancing	Internet of Vehicles
Luo et al. (2020)	✓	BC	✓	Throughput & Energy	Consensus	IoT
Lin et al. (2021)	✓	BC	✓	Block Size & Block generation rules	Privacy Enhancing Tech	Internet of Vehicles
Cao et al. (2022)	✓	BC	✓	Utility	Energy Trading	Energy Internet
DQRL	✓	**DAG**	✓	**Bandwith & Delay & Reliability**	**Path Finding**	**Networks**

Finally, the table includes DQRL, our proposed framework that integrates SDN, DLT (in a DAG-based form), and RL to achieve QoS-driven routing in larger-scale network contexts. By blending centralized control, decentralized consensus, and adaptive learning, DQRL aspires to offer robust and flexible solutions for real-time path finding while maintaining data integrity and security. To our knowledge, the methodology presented here in DQRL is the first to integrate SDN with a DAG-based DLT for a novel network management paradigm driven by QoS metrics in path selection, while also leveraging RL through AS networks.

4. METHODOLOGY

The methodology underlying the proposed DQRL framework is meticulously outlined herein, aiming to enhance routing efficiency within multi-domain SDNs. Through the integration proposed DQRL, the framework effectively addresses inter-AS QoS constraints while ensuring scalability, adaptability, and robust network performance. The subsequent discourse provides a comprehensive exploration of the framework's fundamental components and operational mechanisms.

4.1 Proposed Approach

The proposed DQRL framework proposes an innovative approach for optimizing routing in SDNs. This approach integrates DAG-based DLTwith RL to achieve dynamic, QoS-driven, and scalable routing decisions in multi-domain SDN environments. The framework focuses on addressing inter-AS QoS constraints, enhancing routing efficiency, and maintaining adaptability to network changes.

The DQRL framework consists of several components and methodologies, which are systematically discussed in this section to establish a thorough understanding of the proposed approach. These include the network model, inter-AS path optimization problem, DLT-enabled SDN controller architecture, and the incorporation of DAG structures for transaction validation and QoS management.

Key aspects of the proposed framework include:

- **Network Model**: A representation of an SDN-based network comprising multiple Autonomous Systems (ASes), core devices, border nodes, and inter-AS links.
- **DLT-Adapted SDN Controller**: An advanced SDN controller architecture equipped with modules for blockchain-based transaction management and RL-based routing.

- **Inter-AS QoS-Driven Path Optimization Problem (IA_QdP)**: A mathematical formulation of the routing problem, constrained by multiple QoS parameters such as delay and bandwidth.
- **Pathlet and Transaction Management**: Dynamic definition, assignment, and validation of pathlets using DAG-based DLT structures for secure and efficient routing.
- **RL Integration**: Application of RL techniques, guided by innovative reward functions, to dynamically compute optimal E2E paths.

By integrating these components, the DQRL framework offers significant improvements in routing performance, scalability, and QoS satisfaction compared to existing methods like DRM and PRM. Subsections of this chapter provide a detailed discussion of the system model, the role of DAGs in transaction management, and the RL-based decision-making process. The framework's implementation and evaluation are supported by metrics such as FET, MEE, and other key performance indicators (KPIs).

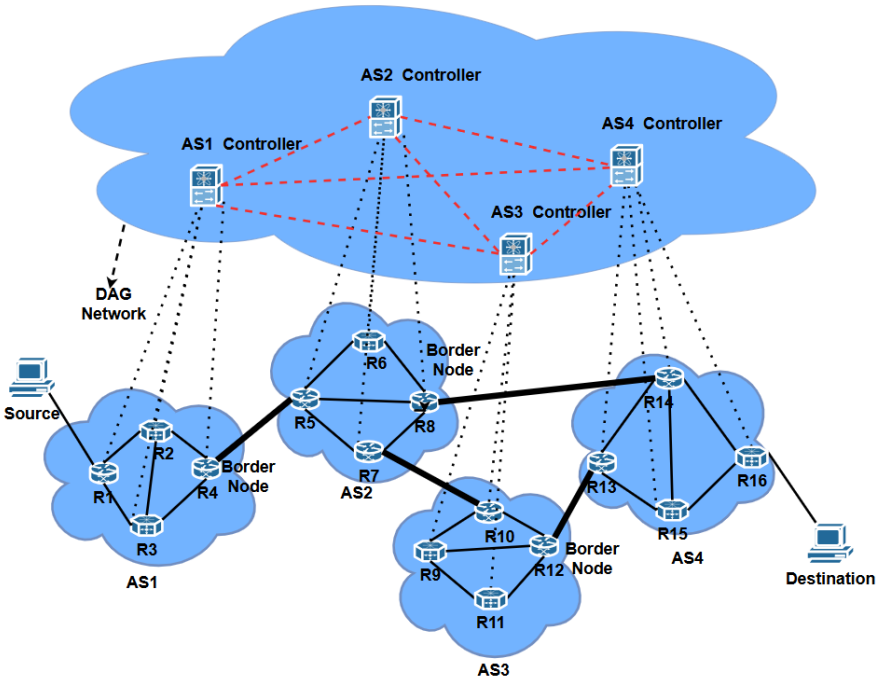
This holistic approach not only ensures robust routing decisions but also highlights the potential of leveraging blockchain and RL to tackle complex network challenges in SDN environments.

4.2 System Model

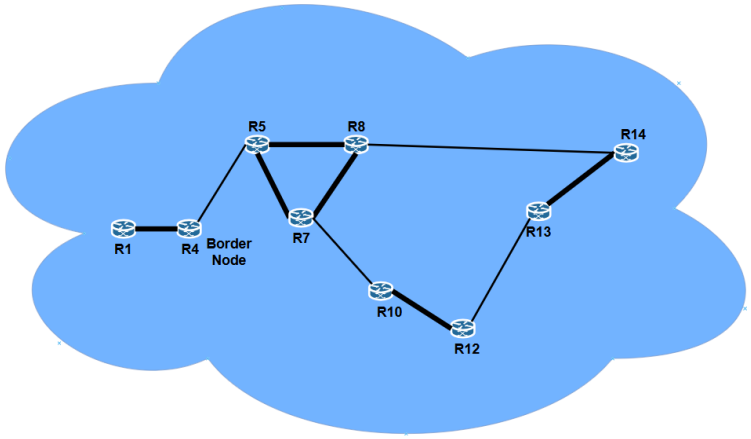
This section introduces a network model designed to implement the proposed DQRL framework.

Figure 12 illustrates a network setup comprising 4 SDN-based Autonomous Systems (ASes) interconnected, with a DAG network connecting the AS controllers with red-dashed lines facilitates the management of network state-related transactions and blocks. The ASes comprise core network components, illustrated as hexagonal shapes, which do not include interconnections, and boundary nodes, depicted as cylindrical shapes, featuring interconnections represented by solid black lines across different ASes. The

control paths linking the controllers to network components are illustrated with dashed black lines.



(a) A sample network model showing 4 SDN ASes.



(b) A layered topology derived from the ASes in Figure 4.1a

Figure 4.1 DAG enabled SDN model and its overlay network

4.2.1 Autonomous systems (ASes)

An Autonomous System (AS) refers to a network that connects geographically distributed communication devices and sub-networks, usually overseen via an Internet Service Provider (ISP). Each AS is granted a unique Autonomous System Number (ASN) by the Internet Assigned Numbers Authority (IANA, 2020), which functions as its unique identifier in inter-AS routing (also called inter-domain routing).

The proposed framework incorporates a private DLTnetwork linking AS controllers (functioning as DLT nodes) to facilitate transaction management and dynamically represent network states. ASes are not required to have physical connections to all other networks. Instead, they maintain inter-AS communication via designated border nodes.

Figure 4.1a demonstrates a network model comprising 4 SDN-based ASes, each managed by a separate controller. Within each AS, network devices are categorized as either:

- Border nodes: Represented as cylindrical objects, these devices connect to other ASes via inter-AS links.
- Core network devices: Represented as hexagonal objects, these devices do not participate in inter-AS connections.

Black dashed lines represent the control pathways that connect controllers to network devices, illustrating the transmission of instructions within the control layer.

Additionally, a DLT-based logical network exists among the AS controllers, depicted by blue dashed lines, which facilitates secure and decentralized coordination among the controllers

4.2.2 DLT- adapted SDN controller

Figure 14 presents a detailed representation of an SDN controller incorporating both traditional and new modules for DLT functionality within the DQRL framework. The

DLT Manager (DM) serves as the DAG-enabling module, comprising various subcomponents responsible for all DAG-related processes within an AS controller. Moreover, the Resource Oversight Module (ROM) consistently assesses network resources, including bandwidth, latency, and jitter. Upon recognizing alterations, the ROM communicates with the DM module to trigger the necessary transactions.

The RL Routing Agent is responsible for handling both inter-AS and intra-AS routing functionalities. By utilizing RL algorithms, it dynamically determines the optimal E2E path for incoming service demands based on real-time QoS metrics such as bandwidth and delay. Unlike static routing approaches, the RL Routing Agent continuously adapts to network changes, ensuring efficient and adaptive routing decisions.

This agent interacts with the DLT Application to retrieve transaction data, including ingress and egress nodes and their respective QoS values, stored in the distributed ledger. By combining RL-based decision-making with the transparency of the DLT framework, the RL Routing Agent enhances routing efficiency while maintaining network integrity.

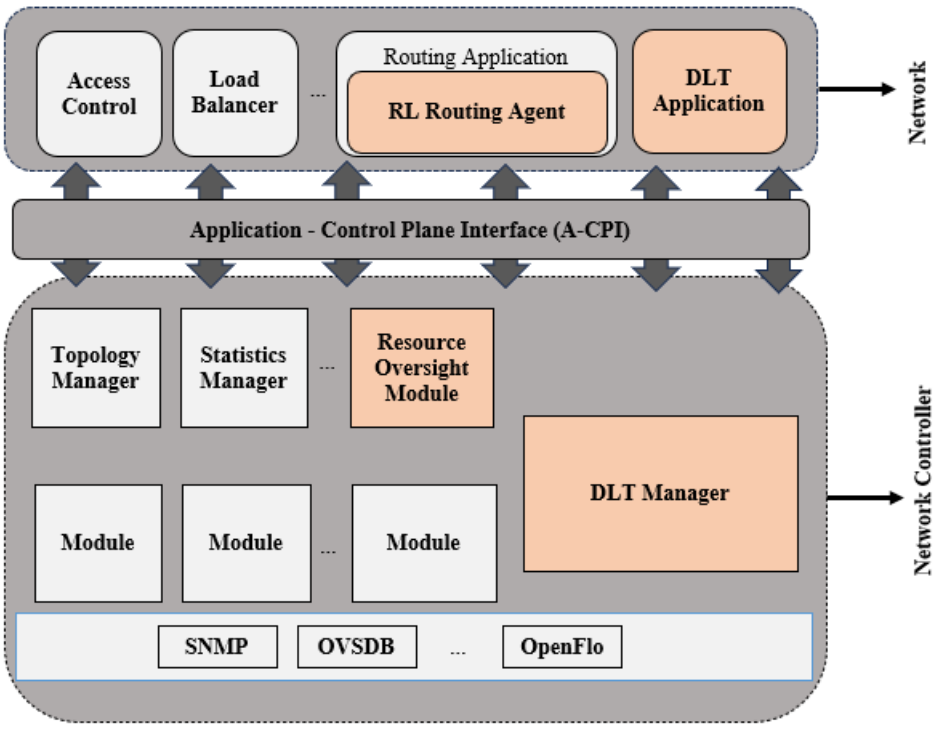


Figure 4.2 DAG-aided SDN controller

4.2.3 DAG structure

A DAG is created for each AS to store the transactions. Each transaction is added to the DAG with the QoS values.

Transactions are added to the DAG, and the QoS values for each pathlet are stored in the DAG.The add_edge () function adds each transaction to the DAG, and each edge contains the QoS values.

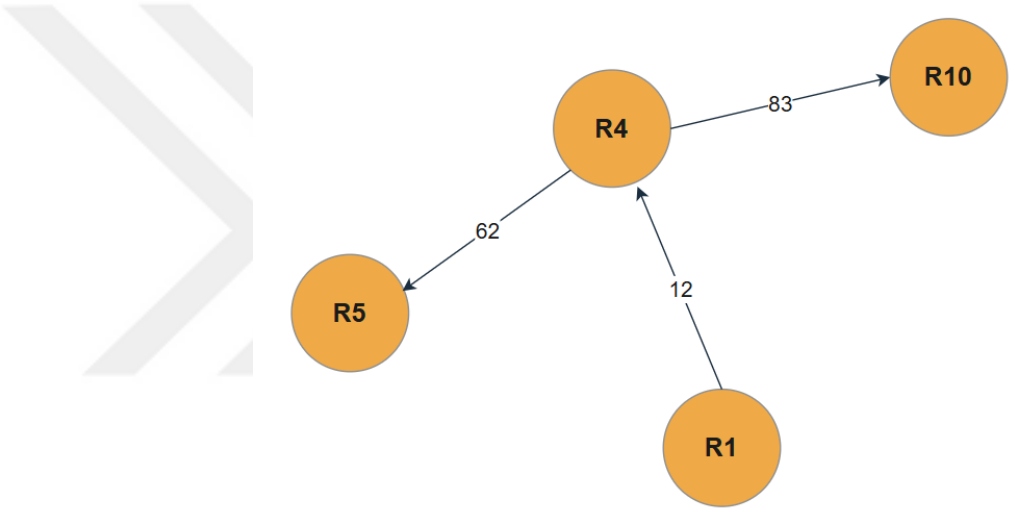


Figure 4.3 DAG-with QoS values

- **Independent Two-Node Validation**: Each transaction necessitates verification by two independent and discrete nodes before being added to the DAG. This ensures the reliability and validity of the DAG.
- **Validation Function:** The validate_transaction () function checks if a transaction is validated by two independent nodes. If there are no two independent nodes, the transaction is considered invalid.
- **Updating QoS Values:** The QoS values for pathlets are periodically updated randomly.
- **Creating New Transactions:** New transactions are created based on the updated QoS values and added to the DAG.

- **Updating Transactions**: The update_qos () function updates the QoS values and creates new transactions. These transactions are validated by two independent nodes before being added to the DAG.
- **Visualizing the DAG**: The DAG for each AS is visualized using NetworkX and Matplotlib. Each edge shows the QoS values for the pathlet.

By following these steps, a DAG-based distributed ledger structure is created for dynamic QoS management and routing decisions in an SDN network. This structure is used to optimize network performance and provide a reliable routing mechanism.

4.2.4 Pathlet and transactions

A pathlet represents all possible paths between a pair of border nodes within an AS. Each pathlet is associated with specific QoS values that are continuously updated.

A pathlet represents all possible paths between a pair of border nodes within an AS, with each pathlet associated with specific QoS values that are continuously updated. Following the formal definition presented in Karakus et al. (2021), a given AS network N_i, with a topology represented as $N_i(V_i, E_i)$, a path $P =< v_i^1, \ldots., v_i^t >$ is called a pathlet if $\forall v_i^j$ in $P) \in V_i$ and $v_i^1, v_i^t \ \in * V_i$, and an edge e_i^l from v_i^j to v_i^{j+1}, $(\forall v_i^j,\ v_i^{j+1} \in P, \forall e_i^l \in E_i)$.

A pathlet can be viewed as a distinct segment of a route that specifically links pairs of boundary devices responsible for network entry and exit. Ingress and egress nodes correspond to the initiating and terminating points of this segment, respectively. Illustratively, Figure 13 shows that the routes R5-R8, R5-R7-R8, and R5-R6-R8 (assuming bidirectional links) serve as pathlets linking network devices R5 and R8 in AS2. Boundary Nodes for ASes: Boundary nodes are established for each Autonomous System (AS). For example: AS1: ['R1', 'R4'], AS2: ['R5', 'R7', 'R8'], AS3: ['R10', 'R12'], AS4: ['R13', 'R14']. *Intra-AS Connections*: Connections within each AS are defined. For example: AS3: [('R9', 'R10'), ('R10', 'R12'), ('R12', 'R11'), ('R11', 'R9'), ('R9', 'R12')].

Inter-AS Border Node Connections: Connections between border nodes of different ASes are defined. For example: [('R4', 'R5'), ('R7', 'R10'), ('R12', 'R13'), ('R8', 'R14')]. All possible paths between border nodes are found using the nx.all_simple_paths () function from the NetworkX library, and these paths are defined as pathlets

4.2.4.1 Assingning QoS values to Pathlets

Each pathlet is assigned random QoS values, including bandwidth (Mbps) and delay (ms). These values are used to determine network performance and quality. Random QoS values are generated for each pathlet using the generate_random_qos () function, which assigns random bandwidth and delay values.

4.2.4.2 Transactions

A transaction records the QoS values for a pathlet. Each transaction captures a specific change in QoS values.

- **Initial Transactions**: To initialize the DAG structure, initial transactions are created for each AS. Each transaction contains the QoS values for a pathlet.
- **Transaction Format**: Each transaction contains the following details:

    - Tx Id: Transaction Identifier
    - Signature: Transaction signature
    - ASN: Autonomous system number
    - Pathlet ID
    - Ingress Switch
    - Egress Switch
    - Max_BW: Maximum bandwidth
    - Min_D: Minimum delay

Table 4.1 Transactions are produced by the AS1 controller

ASN	Tx ID	Signature	Pathlet ID	Ingress Node	Egress Node	Max Bandwidth	Min Delay
AS1	AS1_1	39e3nwxuj3	R1_R4_1	R1	R4	86	7
AS1	AS1_2	lk8wru1xem	R1_R4_2	R1	R4	59	11
AS1	AS1_3	hgnzei3zrr	R1_R4_3	R1	R4	79	14
AS1	AS1_4	wjeyub19tx	R1_R4_2	R1	R4	60	5
AS1	AS1_5	72kz02xrgh	R1_R4_3	R1	R4	70	10
AS1	AS1_6	xuljyljexq	R4_R5_1	R4	R5	46	14

To illustrate the interconnecting link between R4 and R5 within AS1 and AS2, a transaction identified as AS1_6. In a similar manner, transaction AS1_4 updates the pathlet associated with AS1_2 and the R1_R4_2 Pathlet ID. Likewise, transaction AS1_5 modifies the pathlet corresponding to AS1_3 and the R1_R4_3 Pathlet ID.

- Table 4.2 lists the transactions generated by the AS1 controller for pathlets connecting the R1 and R4 border devices in the AS1 network (refer to Fig. 1). The transactions depicted in the table specifically focus on pathlets between R5 and R7. For instance, transaction AS1_6 represents the interconnecting link from R4 to R5 in AS1 and AS2.
- Upon joining the DAG network, an Autonomous System (AS) begins to create initial transactions for the pathlets that connect its border devices, as specified in Table 4.3. These transactions are subsequently transmitted to the pertinent DAG nodes. In the event of a QoS alteration within the network, such as a bandwidth modification that influences a pathlet, the controller generates a new transaction, designated as an update transaction, to represent the changed state of the pathlet.
- Upon joining the DAG network, AS1 initiates the creation of initial transactions containing data for unique pathlets among its border devices, such as R1 and R4. The initial transactions, identified as AS1_1, AS1_2, and AS1_3, correspond to distinct pathlets labeled R1_R4_1 (R1-R2-R3-R4), R1_R4_2 (R1-R3-R2-R4), and R1_R4_3 (R1- R3-R4), respectively.
- The detailed composition of each pathlet (e.g., the complete list of devices traversed) is known exclusively to the controller managing that specific AS (in this case, the AS1 controller). When updates such as bandwidth or delay

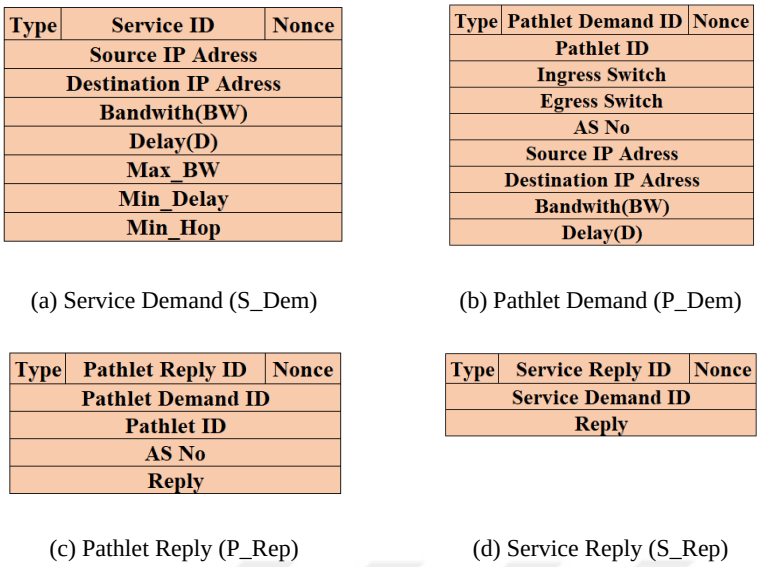
adjustments occur in the network, the controller generates updated transactions to reflect these changes. For example, in Table 4.4, transactions AS1_4 and AS1_5 serve as updates for AS1_2 and AS1_3, respectively, and detail changes in the state of pathlets R1_R4_2 and R1_R4_3, preserving their pathlet IDs.

4.2.5 Service demand

Within the DQRL framework, a Service Demand (SD) signifies the establishment of connectivity with defined QoS requirements between devices, either within the same network or across multiple networks, leveraging routing and switching capabilities. Table 4.1 outlines the abbreviations and their associated definitions used in this framework.

Table 4.2 Abbreviations in the DQRL framework

Acronym	Definition	Acronym	Definition
S_Dem	Service Demand	P_Dem_I	Pathlet Demand Indicators
SNI	Service Network Information	PQI	Pathlet QoS Information
SPC	Service Priority Catalog	P_Dem	Pathlet Demand Information
S_Dem_I	Service Demand Indicators	PNI	Pathlet Network Information
SQI	Service QoS Information	P_Rep	Pathlet Reply Information
S_Rep_I	Service Reply Indicators	PCI	Pathlet Choise Information
SCI	Service Choise Information	P_Rep_I	Pathlet Reply Indicators
S_Rep	Service Reply Information	DSI	Demanded Service Indicators



(a) Service Demand (S_Dem)

(b) Pathlet Demand (P_Dem)

(c) Pathlet Reply (P_Rep)

(d) Service Reply (S_Rep)

Figure 4.4 Structure of the S_Dem, P_Dem, P_Rep, and S_Rep messages.

As illustrated in Figure 4.2a, the structure of an S_Dem message consists of four essential components. The first component, **S_Dem_I**, acts as an identifier for recognizing the SD. This is followed by **SNI**, which provides detailed network-specific information relevant to the SD. Next, **SQI** defines the QoS requirements tied to the SD, and finally, **SPC** emphasizes prioritized parameters for the SD, especially in cases where multiple E2E paths are available.

The S_Dem message also contains several fields. The **Service ID** serves as a unique identifier assigned to the service, while the **Nonce** is a randomly generated value used to ensure the uniqueness of the Service Demand ID. Additionally, the source and destination devices are represented by their respective **IP addresses**. For EtoE connections, the message includes the **bandwidth** and **delay** requirements, referring to the demanded capacity and latency, respectively. Furthermore, **Max_BW** and **Min_Delay** are used to establish the priority among potential paths.

In Figure 4.3b, the P_Dem message is structured similarly, with four main components. The **P_Dem_I** component identifies the Pathlet Demand, while **PNI** supplies network information for the demand. The starting and goal points for the service are defined by

SNI, and the **PQI** component outlines the QoS criterias for the demanded pathlet. The fields within a P_Dem message include a **Pathlet Demand ID**, which uniquely identifies the demand and remains consistent. A **Nonce**, created using a random number, further ensures uniqueness. The **Pathlet ID** marks the specific pathlet being requested from the AS controller. Connection details are defined by the **Ingress Switch** and **Egress Switch**, representing the entry and exit points, respectively. The **AS No** identifies the Autonomous System from which the pathlet is requested, and the source and destination devices are specified by their respective **IP addresses**. Lastly, the bandwidth and delay requirements represent the needed link capacity and latency. These messages are generated by the DLT Application at the source-AS controller and then transmitted to other AS controllers along the chosen E2E path.

Moving on to Figure 4.4c, the P_Rep message in the DQRL framework is composed of three primary parts. **P_Rep_I** functions as the identifier for the Pathlet Reply, while **PNI** provides detailed information about the demanded pathlet. The third part, **PCI**, determines whether the demanded pathlet can be provisioned within the network. The information fields include a **Pathlet Reply ID**, which is both unique and enduring, and a **Nonce**, which is a random value utilized to ensure the uniqueness of the reply ID. The **Pathlet Demand ID** is derived by combining the Pathlet Demand ID and the Nonce sections from the P_Dem message. The **Pathlet ID** distinct identifies the pathlet demanded from the AS controller. Additionally, the **AS No** indicates the originating AS, and the **Reply** reflects the AS controller's decision on whether the demanded connection can be fulfilled.

Finally, Figure 4.5 illustrates the structure of an S_Rep message in the DQRL framework. This information is divided into three main sections. The **S_Rep_I** component identifies the Service Reply, while **DSI** links to the corresponding Service Demand. The **SCI** component reflects the status or outcome of the demanded service. The fields in an S_Rep message include a **Service Reply ID**, which is both unique and persistent, and a **Nonce**, generated as a random value to ensure the distinctiveness of the reply. The **Service Demand ID** is formed by merging the Service ID and Nonce fields from the S_Dem

message. Lastly, the **Reply** conveys the AS controller's decision regarding the demanded service.

4.2.6 RL -based adaptive routing in SDNs

RL is integrated into SDN to dynamically optimize QoS-aware traffic management. In this study, Q-learning is employed to select paths in a DAG-based DLT framework, ensuring optimized routing across multiple ASes.

RL-based routing agent in this network learns from past routing decisions by observing network states, balances delay, bandwidth, and reliability to optimize QoS and leverages a DAG-based DLT for decentralized, validated routing decisions. MDP structure for the RL-based routing is defined as:

- State (s): Represents the network state defined by available pathlets, QoS attributes like delay, bandwidth, reliability and Ingress node.
- Action (a): Selecting a pathlet for forwarding the packet.
- Next State (s'): The egress node of the chosen pathlet, where the agent updates QoS values.
- Reward (R (s, a, s')): A numerical feedback based on QoS performance, guiding the agent towards optimal routes.

The objective is to maximize the cumulative discounted reward over multiple episodes, refining the routing policy dynamically. The RL agent must balance exploration and exploitation when selecting pathlets to ensure both adaptability and long-term optimization.

- Exploration (ϵ probability): The agent randomly selects a pathlet to discover new routes.
- Exploitation (1- ϵ probability): The agent selects the pathlet with the highest Q-value, leveraging learned experiences.

The agent follows an ϵ-greedy policy, where it starts with a high exploration rate (ϵ =1) and gradually reduces it as the model converges toward optimal paths. As training progresses, ϵ decays exponentially to favor exploitation over exploration.

$$\pi(a|s) = \begin{cases} random\ action\ a,\ \ with\ probability\ \epsilon, \\ \\ arg\max_{a} Q(s,a)\ ,\ \ with\ probability\ 1 - \ \epsilon. \end{cases}$$

The reward function is structured to encourage paths that optimize QoS metrics. It is a critical component in RL, enabling the agent to evaluate and learn optimal routing strategies in a dynamic SDN. In this study, the Q-learning-based reward function is designed to optimize QoS-motivated routing, balancing key network metrics such as delay, bandwidth, and reliability

The RL agent interacts with the environment in a state-action-state (*s, a, s′*) framework. Agent's objective is to maximize cumulative rewards over multiple episodes, ensuring that selected paths meet the required QoS constraints while maintaining network efficiency.

Reward Equation:

$$\boldsymbol{R_f(s,a,s') = C_D D_N + C_B B_N + C_R R + C_{TD} DTD_{mth} + [\,C_{Ts} T_s] - S_p} \qquad (2)$$

Where:

- R_f: Reward function.
- C_D: Weight for the delay factor.
- C_B: Weight for the bandwidth factor.
- C_R: Weight for the reliability factor.
- C_{TD}: Negative weight for total delay (penalizes longer delays).
- DTD_{mth}: Accounts for accumulated delays in previous steps.
- T_S: Step Count Penalty**,** penalizes excessive steps taken to reach the destination.
- C_{TS}: Negative weight for total steps taken (penalizes longer paths).
- S_P: Step penalty, a negative value applied at every step.

- D_N: Normalized delay.
- B_N: Normalized bandwidth.
- R : Reliability of the link.

To accommodate different routing requirements, two reward function variations are implemented:

- Delay-Optimized Reward Function (R_{f_d}): $C_D > C_B ; C_D = 8 ; C_B = 4$
  - Assigns higher weight to delay minimization.
  - Used in latency-sensitive applications
- Bandwidth-Optimized Reward Function (R_{f_bw}): $C_B > C_D ; C_D = 4 ; C_B = 8$
  - Prioritizes higher bandwidth paths
  - Used in throughput-sensitive applications.

To ensure a balanced contribution of delay and bandwidth in the reward function, these QoS values are normalized. The delay value is scaled between 0 and 1 using a normalization formula, allowing for a standardized assessment of network performance. This approach ensures that both metrics contribute proportionally to the RL process, preventing any single factor from dominating the reward calculation.

$$\boldsymbol{D_N = 1 - \frac{D}{D_{max}}} \tag{3}$$

where:

- D is the actual delay of the selected pathlet.
- D_{max} is the predefined maximum delay threshold.

Higher values of D_N indicate lower delay, leading to a greater reward. The weighting factor C_D determines how much priority is given to delay minimization. When latency-sensitive traffic is prioritized, a higher C_D is assigned. The normalized bandwidth $\boldsymbol{B_N}$ ensures that the RL agent favors pathlets with higher available capacity:

$$\boldsymbol{B_N = 1 - \frac{B}{B_{max}}} \tag{4}$$

where:

- B is the available bandwidth of the selected pathlet.
- B_{max} is the maximum possible bandwidth.

Higher B_N values mean greater bandwidth availability, increasing the reward. The weighting coefficient C_B controls how much the reward function prioritizes high-bandwidth paths. It is set higher for throughput-sensitive applications.

The reliability metric R evaluates the stability of the chosen pathlet. It is assigned values between 0 and 1, where:

- **1**: Fully reliable pathlet (consistently stable with minimal QoS fluctuations).
- **0**: Unreliable pathlet (frequent failures, inconsistent QoS).

Higher R values indicate more stable paths, leading to higher rewards. C_R is used to adjust the weight of reliability in the routing decision. Highly sensitive networks (e.g., critical infrastructure) require higher C_R values.

The cumulative delay penalty is introduced to discourage paths that experience excessive delays over time. By applying this penalty, the model ensures that routing decisions prioritize low-latency paths, enhancing overall network efficiency and QoS compliance.

$$\boldsymbol{DTD_{mth}} = \frac{\sum_{\boldsymbol{t'=1}}^{\boldsymbol{t}} \boldsymbol{D_{t'}}}{\boldsymbol{D_{max}}} \tag{5}$$

where:

- $\sum_{t'=1}^{t} D_{t'}$ is the total delay accumulated across previous steps.
- D_{max} is the maximum allowed delay.

Higher cumulative delay leads to a lower reward, discouraging high-latency paths. C_{TD} is negatively weighted ($C_{TD} < 0$), ensuring that long-delayed paths are penalized more heavily.

The step count penalty is designed to discourage the selection of paths with an excessive number of steps. By incorporating this penalty into the reward function, the model encourages more efficient routing decisions that minimize unnecessary transitions between nodes. This approach helps maintain lower latency and optimizes overall network performance by favoring shorter and more direct paths.

$$T_S = t$$

Where t is the number of hops taken from source to destination. More steps in the route decrease the reward, encouraging efficient path selection. C_{TS} is negatively weighted $C_{TS} < 0$ to penalize paths requiring unnecessary detours.

The step penalty imposes a small fixed deduction at each step to discourage unnecessarily long routes. It encourages the RL agent to minimize the number of steps taken. By applying this penalty incrementally, the model incentivizes selecting more efficient paths, preventing excessive exploration and reducing overall routing complexity. This approach ensures that the agent prioritizes reaching the destination in the shortest possible steps while maintaining optimal network performance.

$$S_P = 0.1$$

The reward function directly influences Q-learning convergence and path selection.

- Higher C_D leads to low-latency routing.
- Higher C_B prioritizes high-bandwidth paths.
- Penalty terms prevent unnecessary path explorations.
- Normalized QoS metrics ensure fair evaluation.

At convergence, the optimal policy for routing is given by:

$\pi^*(s) = arg\ max_a Q(s,a)$ where $\pi^*(s)$ determines the best path selection strategy.

The Q-learning Algorithm is employed in the DAG-SDN routing framework to enable the RL agent to make optimal routing decisions. In this approach, Q-values are updated iteratively based on the following equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

where:

- $Q(s,a)$: represents the expected reward for taking action aaa in state sss.
- a (Learning Rate): determines the rate at which new experiences override previous knowledge.
- γ (Discount Factor): defines the importance of future rewards.
- R (Reward) is the immediate feedback received after taking action a.
- $max_{a'} Q(s',a')$ represents the estimated best future reward, guiding the agent to select actions that maximize long-term benefits.

The QoS-Aware RL-Based Path Selection Algorithm is designed to train RL agent through multiple episodes, allowing it to explore different routing strategies and gradually refine its decision-making process. By interacting with the network environment, the agent learns to balance delay minimization and bandwidth maximization, ensuring optimal path selection. Over time, the agent adapts to dynamic network conditions, leveraging Q-learning to update its policy based on observed rewards and future state predictions. This iterative learning approach enables the development of an adaptive and efficient routing strategy that optimally meets QoS requirements in SDN environments. Key input parameters include the network graph $G(V,E)$,learning parameters a, γ and ε, the number of training episodes, and the startNode and endNode, which define the routing task. Throughout the training process, the RL agent interacts with the network, learning the most efficient paths by considering QoS constraints. As it iterates over different episodes, it refines its routing policy, ultimately converging to an optimal policy π^* that ensures efficient traffic management by dynamically adapting to network conditions, reducing latency, and improving overall throughput.

The RL model is integrated into a DLT, where each pathlet selection is recorded as a transaction within the DAG. These transactions store QoS values, allowing trustless

validation of routing decisions. The DAG structure supports parallel validation, enhancing scalability and ensuring efficient, real-time network optimization.

Algorithm: QoS-Aware RL-Based Path Selection

```
function RL_Path_Selection(Graph, α, γ, ϵ, episodes, startNode, endNode)
    Initialize Q-values: Q[s, a] ← 0 for all states s and actions a
    for episode ← 1 to episodes do
        s ← startNode
        step ← 0
        totalDelay ← 0
        while s ≠ endNode and step < maxSteps do
            possibleActions ← Filter(Actions[s], minBandwidth, maxDelay)
            if possibleActions = ∅ then break
            # Exploration-Exploitation Tradeoff
            if random() < ϵ then
                a ← RandomChoice(possibleActions)  ▷ Exploration
            else
                a ← argmax Q[s, a] for a in possibleActions  ▷ Exploitation
            # Perform Action and Observe Next State
            delay ← GetDelay(a)
            bandwidth ← GetBandwidth(a)
            reliability ← GetReliability(a)
            totalDelay ← totalDelay + delay
            nextState ← GetEgress(a)
            goalReached ← (nextState = endNode)
            # Compute Reward
            reward ← CalculateReward(delay, bandwidth, reliability,
                                     totalDelay, step, goalReached)
            # Update Q-Table
            Q[s, a] ← Q[s, a] + α * (reward + γ * max Q[nextState, a'] - Q[s, a])
            s ← nextState
            step ← step + 1
            if goalReached then break

        # Decay Epsilon (ϵ)
        ϵ ← max(minEpsilon, ϵ * ϵ_decay)
    end for
end function
```

4.2.7 Service demand workflow of DQRL framework

The operation of the DQRL framework in managing service demands within an SDN network is outlined in Figure 4.6 and explained as follows:

- When an S_Dem message containing a QoS-based Cross-AS SD is provided by a user or client to an AS, the controller initiates the search for an appropriate E2E path. This path comprises pathlets connecting the border node of the source

network to a boundary switch in the destination-AS. The framework utilizes its DAG-based distributed ledger to manage pathlets and QoS-related transactions dynamically, ensuring that network state information remains accurate and current.

- The source-AS controller identifies possible E2E paths using the Ingress Switch, Egress Switch, Max_BW, and Min_Delay attributes within the transactions stored in the DAG ledger. Additionally, RL is leveraged to optimize the path selection process. RL prioritizes paths that fulfill the QoS constraints specified in the S_Dem message by learning from network conditions and adapting dynamically.
- If the source-AS controller is unable to determine an E2E path that fulfills the given QoS criteria through RL-based decision-making, it transmits an S_Rep message to the user or client containing a REJECT response, indicating that the demand cannot be fulfilled. Alternatively, if a suitable E2E path is identified, the controller moves on to the validation phase utilizing the DAG-based ledger. The feasibility of the selected path is verified by querying each AS controller along the path through P_Dem messages (see Figure 16b) to verify their ability to meet the required QoS values.
- Upon receiving the P_dem packets, each AS controller involved in the End to End path replies to the first-AS controller with P_Rep packets (as depicted in Figure 16c). These responses indicate whether the requested pathlet can be provided (ACCEPT or REJECT) with the specified QoS parameters. If a REJECT reply is received, RL adapts by refining the path selection process and considering alternative paths based on prior feedback and rewards.
- If all Ases on the designated path respond with ACCEPT in their P_Rep messages, the source-AS delivers an S_Rep message (illustrated in Figure 16d) to the user or client with an ACCEPT confirmation. This confirmation ensures that the demand is achievable and initiates the service setup within the network. Subsequently, flow entries are installed on the network devices along the E2E path by the respective Ases. Guided by RL, this method enhances efficiency by eliminating redundant assessments of service flows within the ASes.
- If any AS controller along the E2E path replies with a REJECT in their P_Rep message, the source-AS controller starts a new search for an alternative E2E

path. Using RL, the system dynamically assesses other feasible routes while ensuring compliance with the specified QoS constraints. The DAG-based ledger ensures transparency and maintains decision integrity, allowing the framework to adapt seamlessly to changes within the network.

- Intra-domain routing within the source-AS and destination-AS networks is handled using local mechanisms to manage the partial paths. These segments connect the user to the start-AS bourdary switch and the goal-AS bourdary switch to the destination user. Consequently, an E2E path is formed that adheres to QoS requirements and delivers optimal performance by harnessing the integrated capabilities of RL and the DAG-based DLT framework.

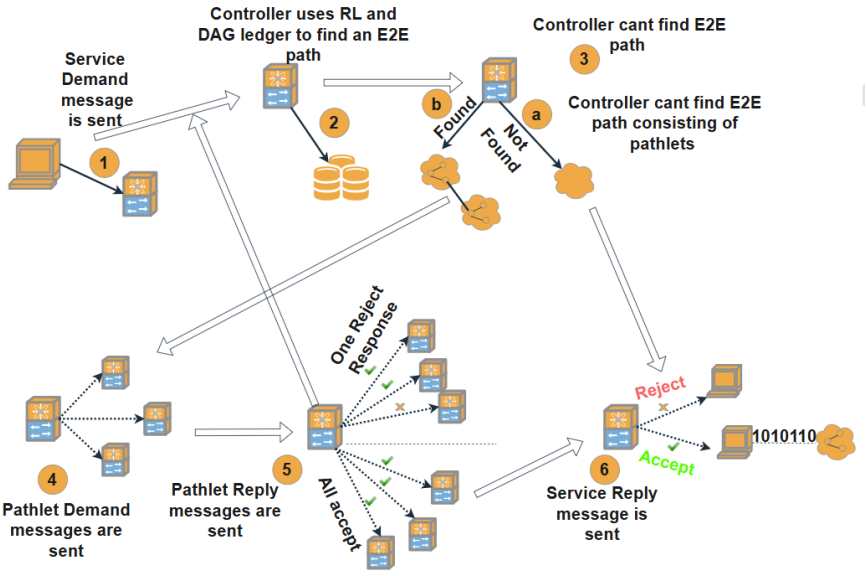


Figure 4.5 Step-by-step procedures in DQRL

5. EVALUATION

This section evaluates the proposed DQRL framework by comparing its performance with traditional routing approaches DRM, PRM, and BCQRM under various network configurations. Each of these frameworks adopts a distinct routing methodology:

- DRM follows a decentralized approach but lacks adaptive learning, leading to suboptimal path selection in dynamic environments.
- PRM improves efficiency through a hierarchical structure; however, it suffers from scalability limitations and increased control overhead.
- BCQRM integrates blockchain for secure QoS-aware routing, but its block-based consensus mechanism introduces validation delays and computational overhead.
- DQRL leverages DAG-based DLT for rapid, parallel transaction validation and RL for adaptive decision-making, ensuring scalable and efficient traffic management.

The evaluation is based on the following key performance metrics: FET, MEE, APL, ANoL, and ANL. These metrics provide a comprehensive basis for comparing the scalability, adaptability, and efficiency of different routing frameworks. The selection of these metrics is motivated by their ability to capture essential network performance characteristics, particularly in dynamic and inter-domain routing scenarios. Similar metrics have been utilized in prior studies on QoS-based inter-AS routing and blockchain-enabled networking solutions (Karakus et al., 2021; Karakus & Guler, 2020).

- FET: Measures the time required to establish an E2E path, reflecting the responsiveness of the routing framework.
- MEE: Quantifies the communication overhead by counting the number of control messages exchanged across the network, indicating the efficiency of signaling mechanisms.

- APL: Represents the efficiency of path selection by measuring the average number of pathlets used in routing decisions, influencing overall network latency and stability.
- ANoL: Determines the total number of nodes traversed in a selected path, providing insight into routing efficiency and network congestion.
- ANL: Measures the average number of AS traversed along an E2E path, reflecting the complexity of inter-domain routing and scalability challenges.

These metrics enable a comprehensive evaluation of the routing frameworks, allowing for a detailed comparison of their performance across different network scenarios. The results highlight the advantages of DQRL, which achieves superior scalability and adaptability by leveraging DAG-based validation and RL-driven learning, outperforming both traditional routing mechanisms and blockchain-based approaches.

5.1 Experimental Setup

SDN Setup: The network topology settings were developed and simulated using the Mininet emulator and the Floodlight controller. These tools facilitated the measurement of parameter values as specified in Table 5.1**.**

Table 5.1 List of parameters and notations

Symbol	Definition	Value (Range)
Swi	The aggregate count of switches configured within the simulation parameters.	*25 - 100*
BRo	Main controller named "Broker"	*1*
B_k	A collection of boundary nodes within the k-th AS	*1 - 4*
N	The collection of ASes configured in the simulation environment.	*5 - 10*
N_g	The collection of ASes that constitute the E2E path calculated for a specific SD_g.	*2 - 6*
C_g	The collection of ASes involved in the E2E path computation for a particular SD_g.	*2 - 6*
Swi_k,g	Number of switches over the E2E path computed for an SD_g in k-th AS	*2 - 7*
T(a)exe_b	The amount of time needed to execute a message (a) within a component (b).	*0.1 - 1 ms*
T(a)broad_b->c	The amount of time needed to broadcast a message (a) from component (b) to component (c).	*8 - 45 ms*
T(path_e2e_dlt)deter_b	The amount of time needed to determine an E2E path for the SD using DLT at controller (b)	*6 - 15 ms*
T(path_L)deter_b	The amount of time needed to determine a local path from a switch to boundary nodes (or viceversa) at controller (b)	*1 - 3 ms*

DLT Setup: A custom DLT network was implemented using Python, incorporating varying numbers of nodes (representing AS controllers) and diverse DAG transaction quantities for the simulations.

Other Setup: The experiments were conducted to evaluate the performance of the proposed framework under different network topologies using both same request and different request scenarios. In the same request scenario, the source and destination switches remain fixed across different topology variations. The network topology is modified in three ways: first, in the fixed-AS (10), varying-SW (5 → 10) inter-domain setup, the number of Autonomous Systems is kept constant at 10 while the number of switches per AS varies from 5 to 10. Second, in the fixed-SW (10), varying-AS (5 → 10) intra-domain setup, the number of switches per AS is fixed at 10, but the number of ASes

varies between 5 and 10. Third, in the varying-SW (5 → 10), varying-AS (5 → 10) inter & intra-domain setup, both the number of switches per AS and the total number of ASes change dynamically.

In the different request scenario, the source and destination switches are randomly selected for each topological variation while applying the same three topology modifications as in the same request scenario. The experiments also account for intra-domain and inter-domain routing by adjusting connectivity probabilities based on the specific topological configuration. In the fixed-AS (10), varying-SW (5 → 10) setup, where the number of ASes remains constant while the number of switches per AS changes, the probability of intra-AS connectivity is set to 0.3, while inter-AS connectivity probability is 0.1. This configuration ensures that the internal structure of each AS remains relatively well connected, whereas inter-domain communication is more constrained. In the fixed-SW (10), varying-AS (5 → 10) setup, where the number of switches per AS is fixed at 10 while the number of ASes varies, intra-AS connectivity probability is reduced to 0.1 while inter-AS connectivity probability is increased to 0.3. This setting reflects a network structure where inter-domain communication plays a more dominant role as the number of ASes increases. Finally, in the varying-SW (5 → 10), varying-AS (5 → 10) inter & intra-domain setup, both intra-AS and inter-AS connectivity probabilities are set to 0.3, creating a more balanced topology where internal AS structures remain well-connected while also maintaining strong inter-domain communication.

Each service demand follows a predefined QoS requirement, where the minimum bandwidth threshold is set to 40 Mbps and the maximum delay threshold is set to 15 ms. Additionally, service demands were initially configured with a fixed bandwidth request of 1 Mbps. To prevent service request rejection due to bandwidth constraints, all network links were assigned a total capacity of 100 Gbps. However, bandwidth availability dynamically decreases based on pathlet usage and the number of iterations. In each iteration, whenever a pathlet is selected for routing, its available bandwidth is reduced proportionally to the predefined minimum bandwidth threshold. If the bandwidth of a pathlet drops below the required threshold, it is no longer considered a viable routing

option. This dynamic bandwidth adjustment mechanism simulates real-world network conditions where frequent usage of a path affects available resources and influences future routing decisions.

To ensure diverse and structured network configurations, a custom topology generator was utilized. Each experiment was executed 20 times, and the results were averaged to achieve a 95% statistical confidence level for both stable and dynamic service demand scenarios. The simulations were conducted on a system running Windows 10 Pro, powered by an Intel Core i7-9750H processor with 16 GB RAM.

5.2 Flow Establishment Time (FET)

Represents the complete duration needed to establish flow rule entries in the flow tables of switches across an E2E path which enables the initiation of a service or flow request. The FET metric is influenced by several factors, broadly categorized as follows:

- **Network/Topology-Based Delays**:
    - Round-Trip Time (RTT) between the Ases and switches.
    - Propagation delay for messages or packets exchanged between controllers.
- **Switch/Controller-Based Delays**:
    - Time taken to process messages or packets at switches and controllers.
    - Path computation time within the controller.

Significant delays in any of these factors can lead to increased flow execution latency, causing performance degradation. A high FET can result in congestion in both the control and data layers, as well as longer failover times. Thus, FET serves as a crucial metric for evaluating the effectiveness and responsiveness of DQRL frameworks in SDN environments.

FET DQRL Equation:

$$FET_{DQRL} = T(S_{Dem})_{exe}^{I_h} + T(S_{Dem})_{broad}^{I_h \to I_{sw}} + T(S_{Dem})_{exe}^{I_{sw}} + T(S_{Dem})_{broad}^{I_{sw} \to I_c} + T(S_{Dem})_{exe}^{I_c} + T\left(Path_{e2e_{dlt}}\right)_{deter}^{I_c} + \max_{\{\forall C \in C_u\}} \{ T(P_{Dem})_{prop}^{I_c \to C} + T(P_{Dem})_{exe}^{C} + T\left(P_{Rep}\right)_{broad}^{C \to I_c} + T\left(P_{Rep}\right)_{exe}^{I_c} + \max_{\{\forall sw \in sw_g\}} \{ T(F)_{prop}^{C \to sw} + T\left(P_{Rep}\right)_{exe}^{I_c} \} \} + T\left(S_{Rep}\right)_{exe}^{I_c} + T\left(S_{Rep}\right)_{broad}^{I_c \to I_{sw}} + T\left(S_{Rep}\right)_{exe}^{I_{sw}} + T\left(S_{Rep}\right)_{broad}^{I_{sw} \to I_h} + T\left(S_{Rep}\right)_{exe}^{I_h} \quad (6)$$

The execution and broadcasting durations of S_Dem and S_Rep messages at or between *I_h(source host)* , *I_sw(source switch)* and *I_c (source controller)* are computed by identifying the maximum execution and broadcasting times of P_Dem and P_Rep messages, as defined by Equation.Additionally, incorporates the flow_mod messages exchanged among controllers and switches along the E2E path.

Since two distinct reward functions are employed—one optimizing for bandwidth efficiency (DQRL_BW) and the other for delay minimization (DQRL_D) the formulation of FET differs accordingly. The use of two separate reward functions results in two different FET values, reflecting the distinct routing decisions influenced by each optimization goal. Specifically, DQRL_BW applies a bandwidth-focused reward function that prioritizes paths with higher capacity, thereby reducing congestion and maximizing throughput. Conversely, DQRL_D follows a delay-centric reward function, favoring routes with lower latency to enhance transmission speed and minimize overall propagation delay. This differentiation ensures that the network dynamically selects paths based on the specific QoS requirements of the traffic, making the routing strategy adaptable to varying service demands.

FET PRM Equation:

$$FET_{PRM} = T\left(P_f\right)_{exe}^{I_h} + T\left(P_f\right)_{broad}^{I_h \to I_{sw}} + T\left(P_f\right)_{exe}^{I_{sw}} + T(P_{in})_{broad}^{I_{sw} \to I_c} + T(P_{in})_{exe}^{I_c} + T(P_{in})_{exe}^{I_{sw}} + (T(M)_{broad}^{I_c \to BRo} + \max_{\{\forall C \in C_g\}} \{ \left[T(M)_{broad}^{BRo \to I_c} + T(PathL)_{deter}^{I_c} + T(M)_{broad}^{I_c \to BRo} \right], \left[T(M)_{broad}^{BRo \to T_c} + T(PathL)_{deter}^{T_c} + T(M)_{broad}^{T_c \to BRo} \right] \} + T\left(Path_{e2e_{dlt}}\right)_{deter}^{BRo} + \max_{\{\forall C \in C_g\}} \{ T(M)_{broad}^{BRo \to C} + T(PathL)_{deter}^{C} + T(M)_{broad}^{C \to BRo} + \max_{\{\forall sw \in sw_g\}} \{ T(F)_{broad}^{C \to sw} \} + T(M)_{exe}^{BRo} \} \quad (7)$$

In a similar manner, the FET for a service demand within the PRM framework is defined in equation. This measure encompasses:

1. The execution and broadcasting times for the $\boldsymbol{P_f}$ and the associated packet inbound message $\boldsymbol{P_{in}}$ between *I_h*, *I_sw* and *I_c*, as described in Equation 6.
2. The longest time observed between:
    - The broadcasting latency for local path demand messages from the Broker (***BRo***) to ***I_c***, along with the internal path determination time at ***I_c***.
    - The same set of operations performed at the target controller (**T_c**).
3. The duration taken by the Broker to identify a suitable E2E path for the service demand.
4. The longest execution and broadcasting durations for path setup messages and flow_mod messages shared between the Broker, controllers, and switches along the E2E path.

FET DRM Equation:

$$\boldsymbol{FET_{DRM} = T(P_f)_{exe}^{I_h} + T(P_f)_{broad}^{I_h \to I_{sw}} + T(P_f)_{exe}^{I_{sw}} + T(P_i)_{broad}^{I_{sw} \to I_c} + T(P_i)_{exe}^{I_c} + T(P_i)_{exe}^{I_{sw}} + \sum_{\forall C \in C_g} \left(T(PathL)_{deter}^{C} + T(M)_{broad}^{C \to Next_c} \right) + \max\{ \sum_{\forall C \in C_g} (T(M)_{exe}^{C} + T(M)_{broad}^{C \to Prev_c}), \max_{\{\forall C \in C_g\}} \{ \max_{\{\forall sw \in Sw_g\}} \{ T(F)_{broad}^{C \to sw} \} \} \}} \quad (8)$$

For the DRM framework, the **FET** for a connection demand **FET_DRM** is described in Equation 8. This includes:

- The execution and broadcast times for the initial packet of the flow *Pf* and its corresponding packet_inbound message *Pi* across ***I_h***, ***I_sw***, and ***I_c***.
- The time required for a controller to execute a local path determination and the broadcast time for decision messages to the subsequent controller along the E2E path.

- The maximum execution and broadcast times for path reservation messages and flow_mod messages exchanged between controllers and switches along the E2E path.

The evaluation focuses on FET as the primary metric to compare different routing frameworks, including DRM, PRM, BCQRM, DQRL_BW, and DQRL_D. The analysis is conducted under varying network conditions, including different switch and AS configurations, to assess the impact of Blockchain and DAG-based approaches on routing efficiency. The comparison also highlights the differences between traditional Dijkstra-based routing and RL-driven adaptive routing.

Figure 5.1a and Figure 5.2b evaluate the effect of increasing the number of switches while maintaining a fixed AS count. In the same request scenario (Figure 5.3a), DRM and PRM exhibit substantial FET fluctuations due to their reliance on precomputed paths and static route selection. DRM, in particular, reaches peak FET values of 262 ms, showing its inefficiency in handling dynamic conditions, while PRM maintains relatively stable but still high values, averaging between 225-240 ms. BCQRM, which integrates Blockchain-based optimizations, achieves better stability than DRM and PRM, with FET values fluctuating between 135-145 ms, yet it remains limited by the constraints of static ledger updates.

Our proposed DQRL framework, represented by DQRL_BW and DQRL_D, demonstrates superior adaptability, maintaining stable FET values around 130-135 ms. The minimal difference between the two models suggests that RL effectively balances bandwidth and delay constraints while leveraging DAG's ability to dynamically update routing decisions. In Figure 5.4b, which presents the different request scenario, DRM performs even worse, with FET peaking at 271 ms due to its lack of adaptability to new source-destination pairs. PRM remains relatively stable at 230 ms, while BCQRM fluctuates around 137-144 ms. DQRL_BW and DQRL_D continue to outperform the other frameworks, reaffirming the efficiency of DAG-based RL routing.

Figure 5.5c and Figure 5.6d assess the impact of increasing the number of ASes while keeping the number of switches fixed. In the same request scenario (Figure 5.7c), DRM experiences a decrease in FET from 222 ms to 174 ms as AS count rises, benefiting slightly from more available inter-domain paths. However, its rigid precomputed path approach prevents optimal route selection. PRM remains stable at approximately 230 ms, further emphasizing its inefficiency in adapting to complex inter-domain traffic.

BCQRM continues to outperform both static models, averaging between 135-141 ms, but its reliance on Blockchain still introduces minor delays due to its sequential transaction validation process. In contrast, DQRL_BW and DQRL_D maintain stable FET values between 132-137 ms, demonstrating the effectiveness of DAG in enabling real-time routing adjustments. Figure 5.8d, which represents the different request scenario, highlights a similar trend. DRM fluctuates between 160-190 ms, struggling to efficiently handle dynamically changing traffic demands, while PRM remains at approximately 230 ms. BCQRM stabilizes around 135-141 ms but remains unable to match the real-time adaptability of DQRL, which maintains the lowest FET values.

Figure 5.9e and Figure 5.10f evaluate network scalability when both AS and switch counts vary simultaneously. In Figure 5.11e (same request scenario), DRM and PRM continue to degrade in performance, with FET values ranging from 168-194 ms for DRM and 226-234 ms for PRM. These results highlight the growing inefficiencies of static routing as network complexity increases. BCQRM provides better stability, averaging 137-146 ms, but still suffers from the limitations of Blockchain's consensus mechanisms, which restrict the speed of route updates.

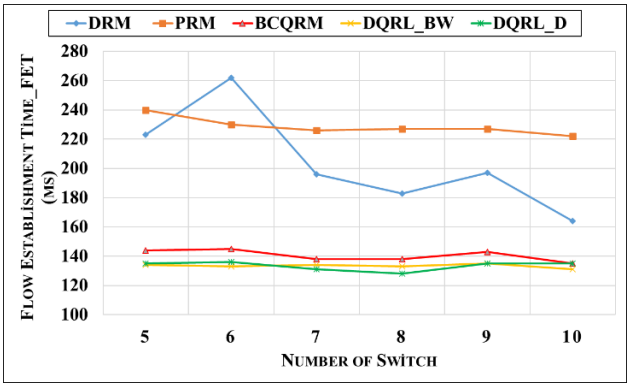
Conversely, DQRL_BW and DQRL_D maintain consistently low FET values, ranging from 130-135 ms, proving that RL-driven routing combined with DAG provides significant advantages in handling both intra- and inter-domain traffic efficiently. Figure 5.12f (different request scenario) reinforces these findings, where DQRL again outperforms all other approaches. DRM and PRM struggle to adapt to increasing network complexity, while BCQRM remains limited by the overhead of Blockchain-based

validation. DQRL benefits from DAG's ability to dynamically update path selections, ensuring optimal route decisions in real time.

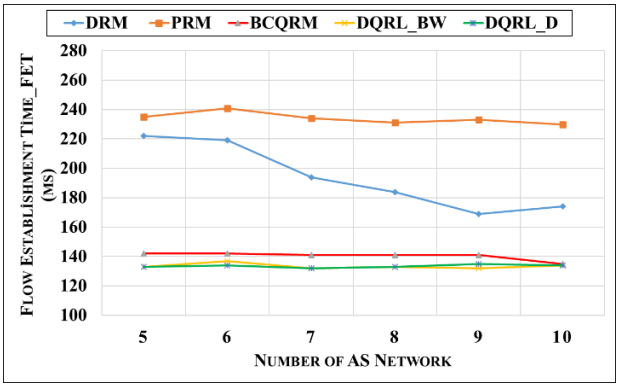
The results clearly indicate that DQRL, leveraging RL and DAG-based decision-making, consistently outperforms traditional routing frameworks. Static routing approaches, such as DRM and PRM, suffer from high FET values due to their reliance on precomputed paths, making them inefficient in large-scale and dynamic environments. BCQRM, while incorporating Blockchain optimizations, still experiences limitations due to its reliance on sequential ledger updates and consensus-based validation.

In contrast, DQRL_BW and DQRL_D achieve the lowest FET values across all tested scenarios. Their ability to dynamically adapt routing decisions based on real-time network conditions makes them highly scalable and efficient. The integration of DAG enhances this adaptability by enabling decentralized, parallelized updates, reducing latency and improving responsiveness.

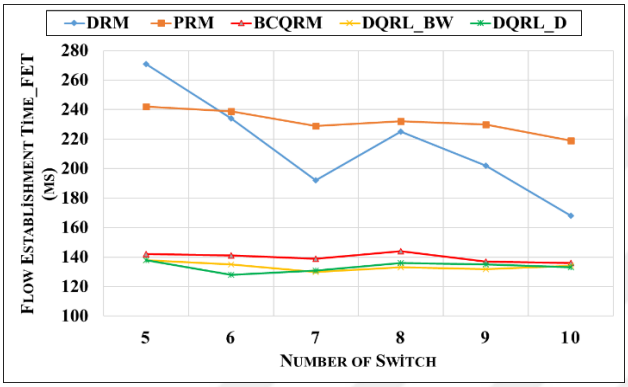
As networks grow in size and complexity, the inefficiencies of traditional routing frameworks become more pronounced. RL-based models, particularly when combined with DAG, provide a practical and effective solution by continuously optimizing routing decisions based on real-time data. These findings demonstrate that DQRL, with its RL-driven and DAG-enhanced architecture, is the superior framework for managing traffic in dynamic, large-scale SDN environments.



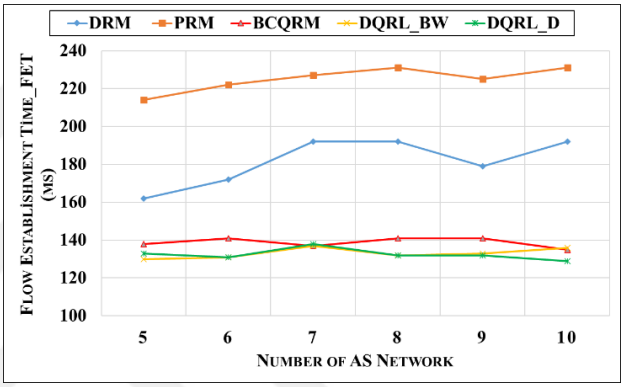
a) Same Request - Fixed-AS (10), Varying SW (5 - 10)



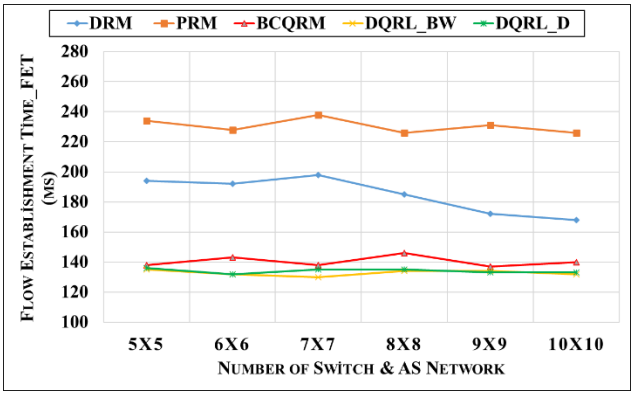
c) Same Request - Fixed-SW (10), Varying AS (5 - 10)



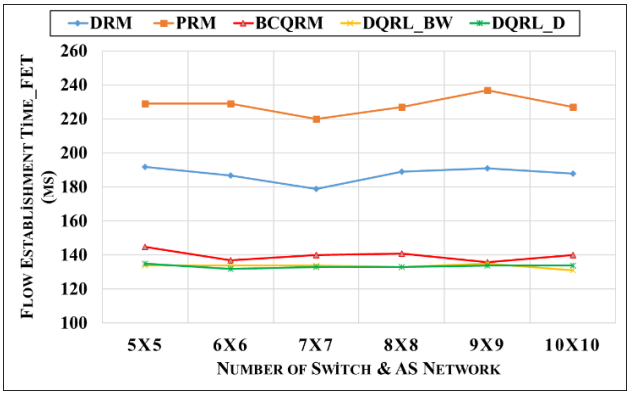
b) Different Request - Fixed-AS (10), Varying SW (5 - 10)



d) Different Request - Fixed-SW (10), Varying AS (5 - 10)



e) Same Request - Varying AS (5 - 10) , Varying SW (5 - 10)



f) Different Request - Varying AS (5 - 10) , Varying SW (5 - 10)

Figure 5.1 FET comparison of DQRL, DRM, and PRM with Dijkstra and RL

5.3 Messages Exchanged and Executed (MEE)

In SDN, MEE denote the control messages shared between controllers and network devices to configure E2E paths that satisfy QoS criteria for service or flow requests. As the network size and the number of service requests increase, the frequency of these messages rises, leading to elevated communication and network overhead.

The MEE metric is directly correlated with this overhead, as it includes all message exchanges required for path setup, such as path computation, flow rule installation, and QoS-related updates. Excessive message exchange can result in significant processing delays and create bottlenecks at controllers, which have limited computational tools.

Minimizing the MEE is essential for improving:

- Controller Performance: Reducing computational load and processing delays.
- Routing Efficiency: Accelerating flow setup processes.
- Scalability: Ensuring smooth operation as network size and service requests grow.

MEE acts as a key indicator for evaluating the effectiveness, scalability, and overall efficiency of routing frameworks in SDN environments.

The number of MEE to set up an E2E path for a service/ flow request in DQRL framework (MEE_{DQRL}), PRM framework (MEE_{PRM}), and DRM framework (MEE_{DRM}) can be given as in Eqs. (8) – (10), respectively.

The number of MEE required to establish an E2E path for a service or flow request within the DQRL framework (MEE_{DQRL}), PRM framework (MEE_{PRM}), and DRM framework (MEE_{DRM}) can be given below, respectively

MEE DQRL Equation:

$$\boldsymbol{MEE_{DQRL} = 2 \times (|C_g|) + \sum_{\forall C \in C_g} (sw_g)} \quad (9)$$

MEE primarily involves the following steps:

- The initiating host (*I_h*) sends an S_Dem message to the initiating controller (*I_c*).
- The *I_c* forwards P_Dem messages to other controllers located along the designated E2E path for the service request.
- Controllers on the E2E path reply to the P_Dem messages by sending P_Rep messages back to *I_c*.
- The *I_c* responds to *I_h* with an S_Rep message.
- flow_mod messages are distributed from controllers to their corresponding switches across the E2E path within their respective networks.

Similar to the FET evaluation, MEE is analyzed in two different ways within the DQRL framework due to the use of two distinct reward functions: DQRL_BW and DQRL_D. These variations allow for a more comprehensive assessment of network efficiency, where DQRL_BW focuses on optimizing bandwidth utilization, while DQRL_D prioritizes minimizing delay. This distinction ensures that both bandwidth-constrained and delay-sensitive scenarios are effectively evaluated within the DQRL model.

MEE PRM Equation:

$$\boldsymbol{MEE_{PRM} = 4 + (|BRo^{I_c}|) + (|BRo^{T_c}|) + [2 \times (|C_g|)] + \sum_{\forall C \in C_g} (sw_g)} \quad (10)$$

As described in Equation, MEE for the PRM framework includes a packet_inbound message being sent from the initiating switch (*I_sw*) to the initiating and target controllers (*I_c* and *T_c*) to advertise local paths between border nodes and source/destination switch pairs. Following this, the source controller sends an E2E path request message to the controller broker (Bro). The broker processes two computation requests and receives

pathlet advertisements from the source and destination controllers for the respective border nodes and switch pairs. It then communicates border node pairs in each network to the controllers on the E2E path. Controllers send confirmation messages for the requested pathlets back to the broker, and flow_mod messages are transmitted from the controllers to the switches in their respective networks to finalize the path setup.

The MEE in the PRM framework involves the transmission of a packet_inbound message from the initiating switch (*I_sw*) to both the initiating and target controllers (*I_c and T_c*). This message advertises local paths between border nodes and the source/destination switch pairs. Subsequently, the source controller sends an E2E path request to the broker (Bro). The broker handles two determination requests and collects pathlet advertisements from the source and destination controllers for the respective border nodes and switch pairs. Afterward, it distributes information about boundary node pairs in each network to the controllers along the E2E path. In response, the controllers send confirmation messages for the demanded pathlets to the broker, and flow_mode messages are dispatched from the controllers to their respective switches, completing the path setup process.

MEE DRM Equation:

$$\boldsymbol{MEE_{DRM}} = \mathbf{1} + \left[\mathbf{2} \times \left(\left|\boldsymbol{C_g}\right|\right)\right] + \sum_{\forall \boldsymbol{C} \in \boldsymbol{C_g}} \left(\boldsymbol{sw_g}\right) \qquad (11)$$

As outlined in Equation, the MEE process in the DRM framework involves the initiating switch (*I_sw*) sending a packet_inbound message to the initiating controller (*I_c*). Following this, controllers along the E2E path exchange decision and notification messages to reserve the required path for the demanded service. Finally, controllers transmit flow_mode messages to the switches in their respective networks, finalizing the setup and configuration of the path for service delivery.

The MEE metric is analyzed to assess the efficiency of different routing frameworks, including DRM, PRM, BCQRM, DQRL_BW, and DQRL_D, under various network conditions. Following the formulation of the MEE metric, this evaluation compares

traditional Dijkstra-based routing methods with RL-based adaptive routing to highlight the impact of Blockchain and DAG structures on control message overhead and overall routing efficiency. The analysis provides insights into how these frameworks manage network dynamics, demonstrating the effectiveness of DAG-integrated RL models in reducing message exchanges while maintaining optimal routing performance.

Figure 5.13a and Figure 5.14b evaluate the effect of increasing switch counts while maintaining a fixed AS count. In the same request scenario, DRM and PRM exhibit relatively high MEE values due to their reliance on frequent controller-to-controller communications for route calculations. PRM performs worse than DRM, peaking at 23 exchanged messages, while DRM maintains values between 15-20 messages as switch count increases. BCQRM, which integrates Blockchain for transaction validation, achieves a moderate reduction in MEE compared to PRM and DRM, but still incurs additional messaging overhead due to ledger synchronization.

In contrast, our proposed DQRL framework, represented by DQRL_BW and DQRL_D, significantly reduces message exchange, maintaining 9-12 messages on average. This reduction is attributed to DAG's ability to update routing decisions asynchronously and in parallel, avoiding the sequential validation constraints seen in BCQRM. In the different request scenario (Figure 5.15b), the inefficiency of traditional methods becomes more evident. DRM and PRM increase their messaging overhead to accommodate dynamically changing source-destination pairs, leading to peaks of 24 messages for PRM and 21 for DRM. BCQRM stabilizes between 15-20 messages, still struggling with the constraints of its consensus mechanism. DQRL maintains its efficiency, keeping MEE values consistently lower, demonstrating its scalability in dynamic network conditions.

Figure 5.16c and Figure 5.17d assess the impact of increasing AS counts while keeping the number of switches fixed. In the same request scenario, DRM and PRM show a decreasing trend in MEE values as additional ASes provide more routing alternatives, reducing the need for extensive controller interactions. However, both methods still suffer from high overhead, with PRM maintaining values between 22-25 messages and DRM stabilizing around 13-22 messages. BCQRM achieves moderate reductions in messaging

overhead due to its Blockchain-based optimizations but still reaches 18 messages at its peak.

DQRL_BW and DQRL_D, leveraging RL and DAG, maintain the lowest MEE values across all AS counts, fluctuating between 9-12 messages. The DAG structure allows faster transaction validation and route adjustments without requiring excessive control message exchanges. In the different request scenario (Figure 5.18d), the same trend is observed, but DRM and PRM show increased instability due to dynamic path recalculations. PRM reaches 21 messages, while BCQRM fluctuates between 14-17 messages, demonstrating its relative inefficiency in handling real-time topology updates. DQRL-based models remain stable and efficient, ensuring minimal control overhead while adapting to changing network conditions.

Figure 5.19e and Figure 5.20f analyze scalability by varying both AS and switch counts simultaneously. In the same request scenario, DRM and PRM continue to exhibit high MEE values, ranging from 16-22 messages, reflecting their dependency on precomputed paths and increased inter-domain communication. BCQRM performs slightly better but remains limited by its sequential Blockchain-based ledger updates, keeping MEE values between 12-16 messages.

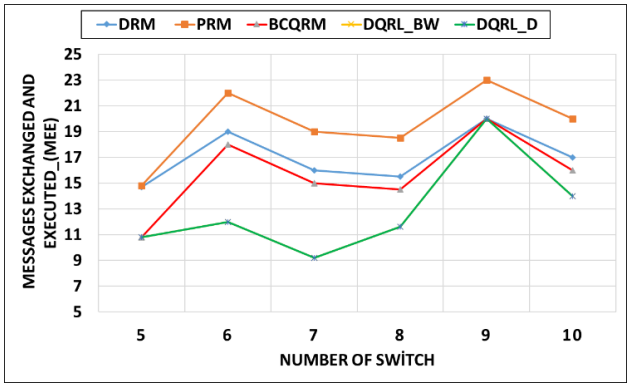
Conversely, DQRL_BW and DQRL_D consistently maintain the lowest MEE values, between 8-10 messages, proving the advantages of RL-driven routing combined with DAG. The parallelized transaction updates within the DAG structure minimize the need for frequent control message exchanges, enhancing efficiency. In the different request scenario (Figure 5.21f), DRM and PRM perform significantly worse, with PRM reaching 25 messages at its peak and DRM fluctuating between 17-22 messages. BCQRM stabilizes around 16-21 messages, but it remains inferior to DQRL. The RL-DAG integration in DQRL enables real-time adaptation, keeping MEE values at a minimum while ensuring stable routing performance.

The analysis of MEE demonstrates that DQRL, leveraging RL and DAG-based routing, consistently minimizes control message overhead compared to traditional routing

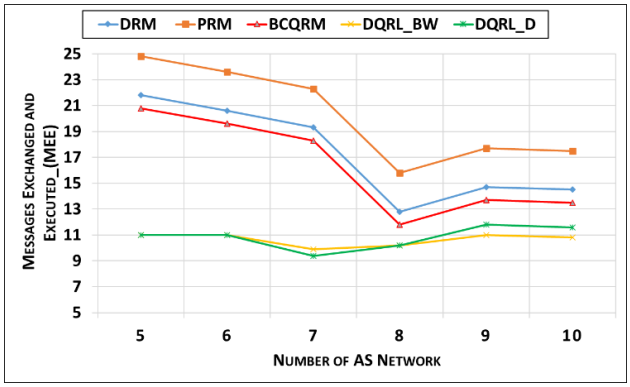
frameworks. DRM and PRM, relying on precomputed paths and static route calculations, struggle to maintain low MEE values as network complexity increases. BCQRM, though incorporating Blockchain optimizations, still faces challenges due to its reliance on sequential transaction validation, which increases message exchange requirements.

In contrast, DQRL_BW and DQRL_D achieve the lowest MEE values across all tested scenarios, highlighting their efficiency in reducing control message exchanges while maintaining high adaptability. The DAG structure enhances this advantage by enabling decentralized, asynchronous updates, reducing overhead and improving responsiveness. As network size and complexity grow, traditional frameworks become increasingly inefficient, while RL-based models, particularly when integrated with DAG, provide a scalable, real-time routing solution.

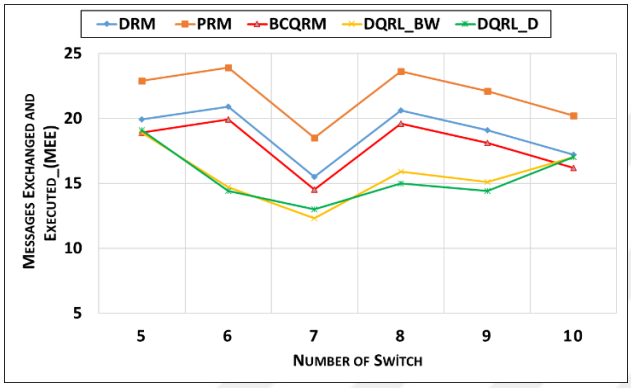
This evaluation confirms that DQRL, with its RL-driven and DAG-enhanced approach, is the most efficient framework for minimizing control message overhead while ensuring optimal routing decisions in large-scale SDN environments. The findings further suggest that DAG-based architectures offer a compelling alternative to Blockchain-based methods in terms of reducing network overhead and improving responsiveness in dynamic, multi-AS networks.



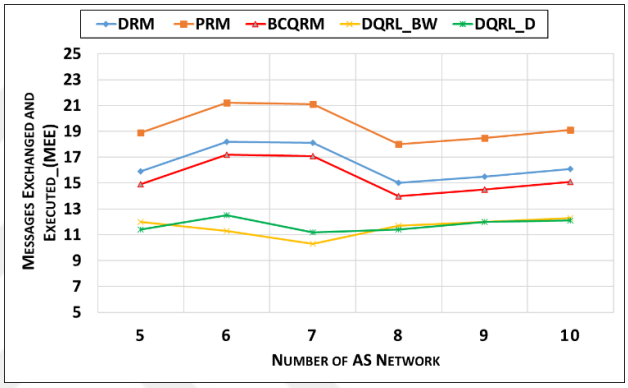
a) Same Request - Fixed-AS (10), Varying SW (5 - 10)



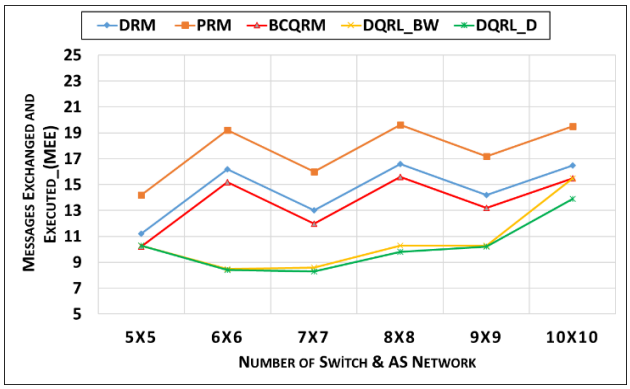
c) Same Request - Fixed-SW (10), Varying AS (5 - 10)



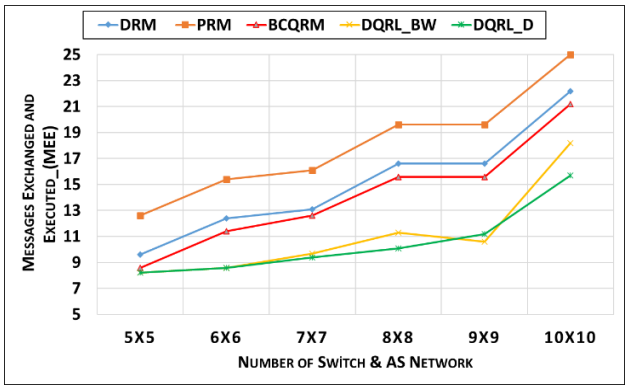
b) Different Request - Fixed-AS (10), Varying SW (5 - 10)



d) Different Request - Fixed-SW (10), Varying AS (5 - 10)



e) Same Request - Varying AS (5 - 10) , Varying SW (5 - 10)



f) Different Request - Varying AS (5 - 10) , Varying SW (5 - 10)

Figure 5.2 MEE comparison of DQRL, DRM, and PRM with Dijkstra and RL

5.4 Average Pathlet Length (APL)

Average Pathlet Length refers to the average number of switch contained in a pathlet. A pathlet represents a subpath connecting two border nodes, and this metric calculates the average node count in the paths used to fulfill service requests. APL as a critical metric for assessing the efficiency of routing approaches in SDN environments. APL is defined as the ratio of the total number of switches traversed to the total number of pathlets (P), formulated as:

APL Equation:

$$\boldsymbol{APL} = \frac{\sum \boldsymbol{sw}}{\sum \boldsymbol{p}} \tag{12}$$

$\sum \boldsymbol{sw}$ represents the total number of switches in the network and $\sum \boldsymbol{p}$ denotes the number of selected pathlets. A lower APL value indicates that a routing approach effectively minimizes unnecessary switch traversals, leading to more direct and efficient paths. Conversely, higher APL values suggest that packets travel through additional intermediary switches, potentially increasing latency and network overhead.

The evaluation of APL examines the effectiveness of different routing methodologies, specifically comparing Dijkstra-based routing with RL-based approaches, RL_BW and RL_D. By analyzing how path lengths evolve under varying network conditions, this study highlights the efficiency of RL in optimizing routing decisions and minimizing path overhead.

Figure 5.22a and Figure 5.23b evaluate how increasing the number of switches affects APL while keeping the AS count fixed. In the same request scenario , Dijkstra-based routing exhibits consistently higher APL values, fluctuating between 3.0 and 3.6, indicating longer path usage due to its reliance on precomputed routes. As switch density increases, RL_BW and RL_D maintain significantly lower APL values, averaging between 1.7 and 2.0, demonstrating their adaptability in selecting efficient routing paths dynamically.

In the different request scenario, Dijkstra's inefficiency is further emphasized, with an initial peak of 4.7 APL when the switch count is low. As the number of switches increases, APL values for Dijkstra remain above 3.0, while RL-based methods consistently maintain APL within the range of 2.4 to 2.9, highlighting their ability to adjust to varying source-destination pairs.

Figure 5.24c and Figure 5.25d analyze how increasing the number of ASes while keeping the switch count constant affects APL. In the same request scenario, Dijkstra's APL decreases from 3.6 to 2.1, as additional ASes provide more inter-domain routing alternatives. However, its reliance on static routing keeps APL values high compared to RL-based methods. RL_BW and RL_D sustain APL values between 1.7 and 2.0, ensuring efficient path selection while maintaining stability.

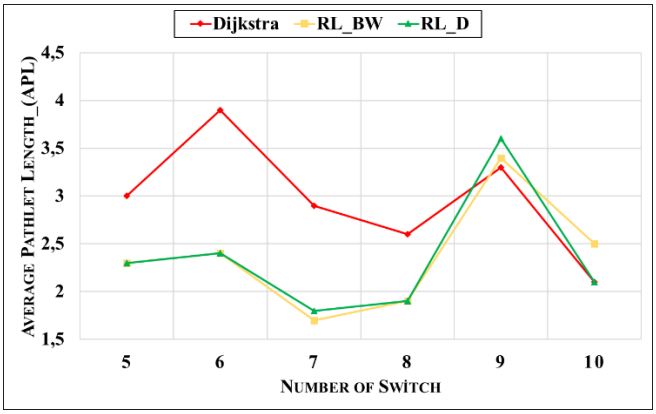
In the different request scenario, the performance gap becomes more evident. Dijkstra's APL fluctuates between 2.4 and 2.8, whereas RL_BW and RL_D maintain lower pathlet lengths, consistently ranging between 1.8 and 2.0. These results highlight the limitations of traditional static routing methods in adapting to dynamic changes in AS topology, whereas RL-driven models effectively optimize path selection in real time.

Figure 5.26e and Figure 5.27f evaluate scalability by varying both AS and switch counts simultaneously. In the same request scenario, Dijkstra exhibits considerable fluctuations, peaking at 3.0 APL, while RL_BW and RL_D maintain more stable values, fluctuating between 1.6 and 2.4, reinforcing their efficiency in dynamically adjusting to network changes.

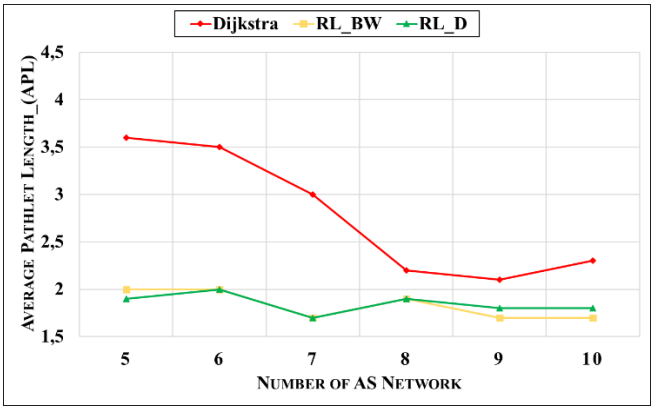
In the different request scenario, Dijkstra's APL fluctuates between 1.9 and 3.0, further proving its inefficiency in handling large-scale network adjustments. RL_BW and RL_D continue to minimize pathlet lengths, ensuring optimal route selection and maintaining APL values between 1.6 and 2.5 across all scenarios.

The analysis of APL in Figure 5.28 demonstrates that RL-based routing methods significantly outperform Dijkstra in reducing path length across different network scenarios. Dijkstra's reliance on precomputed paths results in inefficient routing, leading to longer APL values, particularly in dynamic request scenarios. RL_BW and RL_D, leveraging RL, dynamically adjust paths based on real-time network conditions, consistently maintaining lower APL values.

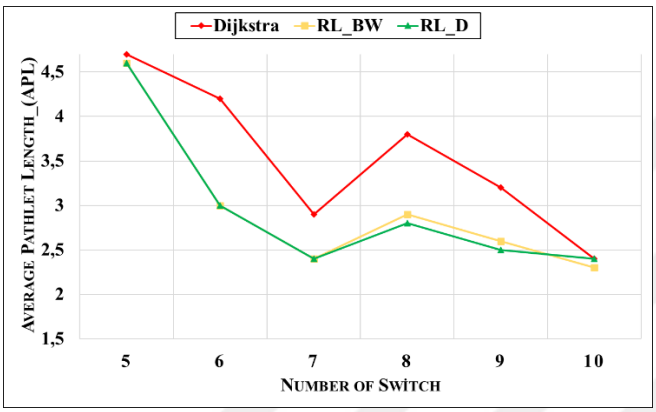
Furthermore, the results highlight that RL-based models not only optimize path selection but also improve overall network efficiency by reducing unnecessary pathlet usage. This is particularly crucial in large-scale SDN environments, where minimizing path length contributes to enhanced QoS and network performance. As a result, RL-driven routing emerges as a more effective approach than traditional Dijkstra-based methods, offering better adaptability, shorter routing paths, and improved overall efficiency.



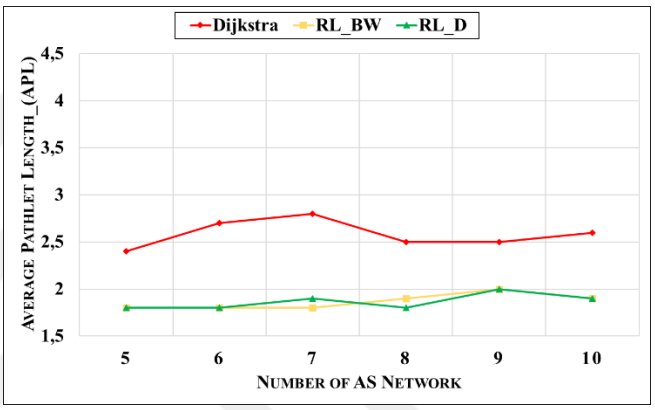
a) Same Request - Fixed-AS (10), Varying SW (5 - 10)



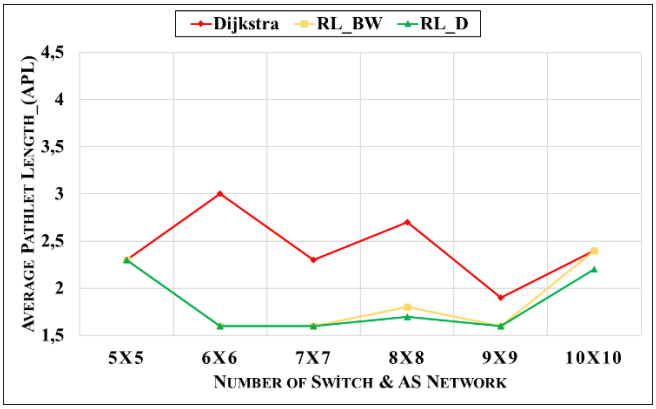
c) Same Request - Fixed-SW (10), Varying AS (5 - 10)



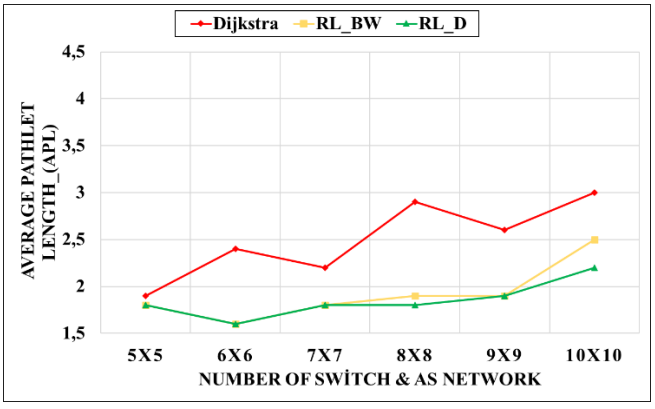
b) Different Request - Fixed-AS (10), Varying SW (5 - 10)



d) Different Request - Fixed-SW (10), Varying AS (5 - 10)



e) Same Request - Varying AS (5 - 10) , Varying SW (5 - 10)



f) Different Request - Varying AS (5 - 10) , Varying SW (5 - 10)

Figure 5.3 APL comparison of Dijkstra and RL_BW and RL_D

5.5 Average Node Length (ANoL)

This metric quantifies the average "hop count" or the average number of nodes that an agent traverses in successfully reaching the destination. It provides a measure of the efficiency of the routing decisions: a lower value indicates that, on average, the agent is selecting more direct routes (fewer intermediate nodes), whereas a higher value implies that the agent's chosen paths tend to be longer.

Let:

- R_s is the total number of successful runs. It means where a complete path from start to goal was found.
- P_i be the path discovered during the i-th successful run.
- $|P_i|$ denote the number of nodes visited on that path (including both the start and goal nodes).

$$\boldsymbol{Avg_{No_L}} = \frac{\mathbf{1}}{\boldsymbol{R_s}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{R_s}} \boldsymbol{|P_i|} \tag{13}$$

This equation indicates that the average node count is obtained by summing the lengths (in nodes) of all successful paths and then dividing by the total number of such paths. This measure effectively captures the overall efficiency of the routing process.

The following subsections provide a comparative evaluation of ANoL across different network scenarios. The assessment of ANoL investigates the efficiency of different routing approaches, specifically comparing Dijkstra, RL_BW, and RL_D under varying network configurations. By analyzing how the number of nodes traversed per path changes with different topology conditions, this evaluation demonstrates the adaptability of RL-based routing compared to traditional static path computation methods.

Figure 5.29a and Figure 5.30b analyze the effect of increasing switch counts on ANoL while keeping the number of ASes constant. In the same request scenario (Figure 5.31a),

Dijkstra-based routing demonstrates higher ANoL values, fluctuating between 3.1 and 4.9, showing a longer average node traversal due to its reliance on precomputed paths. As the switch count increases, RL_BW and RL_D maintain more efficient node selection, stabilizing within the range of 2.7 to 3.5, emphasizing their ability to adapt dynamically.

In the different request scenario (Figure 5.32b), Dijkstra's inefficiency is more pronounced, reaching an ANoL peak of 5.7 when the number of switches is low. Although it decreases as switch count rises, it remains above 3.4, whereas RL-based models maintain significantly lower node traversal values, consistently between 3.3 and 4.0. This result reinforces the advantage of RL-based dynamic routing in optimizing path selection, particularly when network traffic patterns are not predetermined.

Figure 5.33c and Figure 5.34d evaluate the effect of increasing AS numbers while keeping the switch count fixed. In the same request scenario (Figure 5.35c), Dijkstra's ANoL decreases from 4.6 to 3.3 as additional AS connections allow more flexible inter-domain routing. However, its reliance on static routing causes inefficiencies compared to RL-based models. RL_BW and RL_D maintain significantly lower ANoL values, ranging from 2.7 to 3.0, ensuring more efficient node utilization.

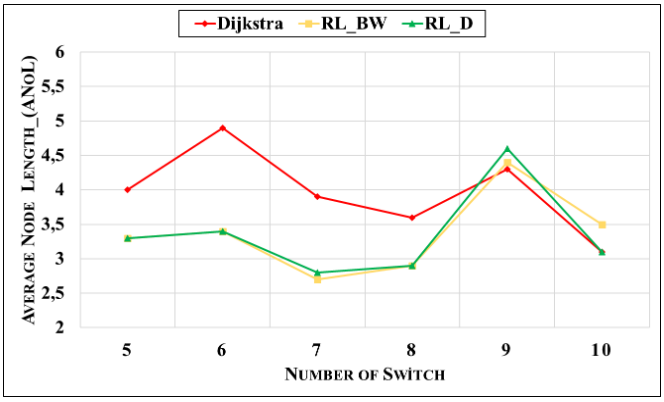
In the different request scenario (Figure 5.36d), the gap between static and adaptive routing becomes more apparent. Dijkstra's ANoL fluctuates between 3.4 and 3.8, whereas RL_BW and RL_D demonstrate more efficient node usage, stabilizing between 2.8 and 3.0. These results emphasize that RL-based routing is better suited to handling dynamic AS environments, whereas static routing struggles to adapt to topological changes.

Figure 5.37e and Figure 5.38f assess how ANoL scales when both AS and switch counts are varied. In the same request scenario (Figure 5.39e), Dijkstra's ANoL fluctuates significantly, reaching 3.7 at its peak, while RL_BW and RL_D maintain consistently lower values, ranging from 2.6 to 3.4. The RL-based models efficiently adjust to changes in network topology, preventing unnecessary node traversals.

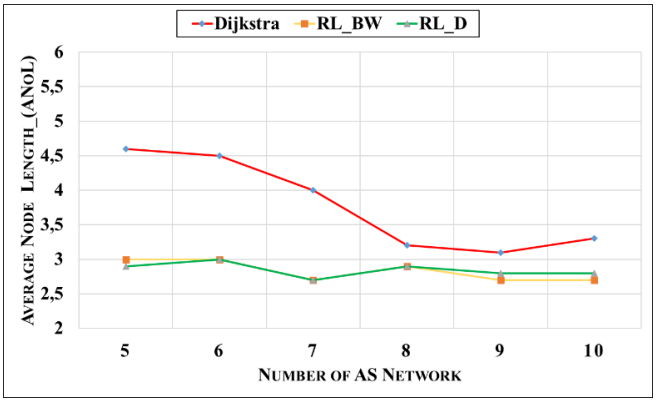
In the different request scenario (Figure 5.40f), Dijkstra exhibits increased variability, with ANoL values ranging from 2.9 to 4.0, further highlighting its limitations in adapting to large-scale network variations. In contrast, RL_BW and RL_D sustain lower node traversal values between 2.6 and 3.5, reinforcing their efficiency in selecting optimal paths dynamically.

The ANoL analysis in Figure 5.41 confirms that RL-based routing significantly outperforms Dijkstra in minimizing node traversal across different network configurations. Dijkstra's static approach leads to inefficient path selection, resulting in higher ANoL values, particularly in dynamic network conditions where the topology is not precomputed.

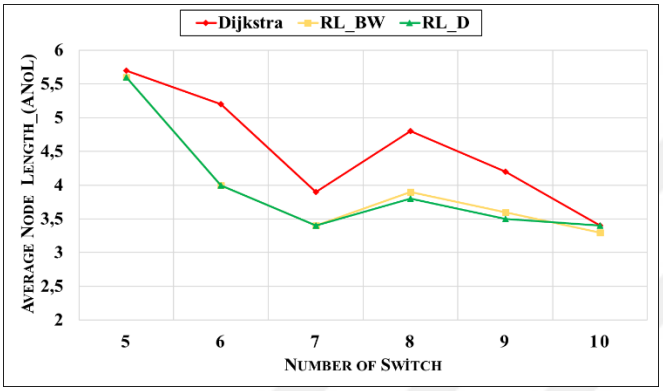
In contrast, RL_BW and RL_D consistently reduce node traversal by adapting to real-time network states, ensuring shorter and more efficient paths. The findings suggest that RL-driven routing not only optimizes node utilization but also enhances overall network efficiency by reducing unnecessary node hops, leading to improved QoS and better scalability in SDN environments.



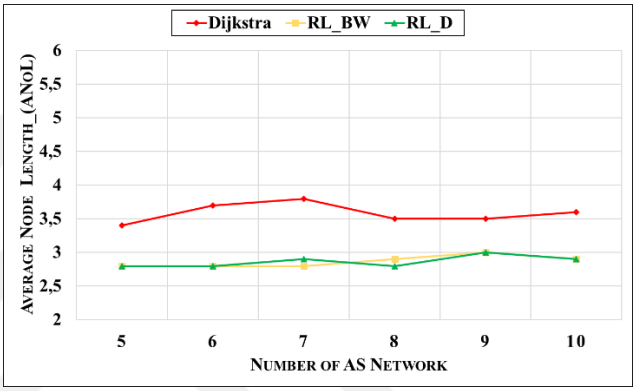
a) Same Request - Fixed-AS (10), Varying SW (5 - 10)



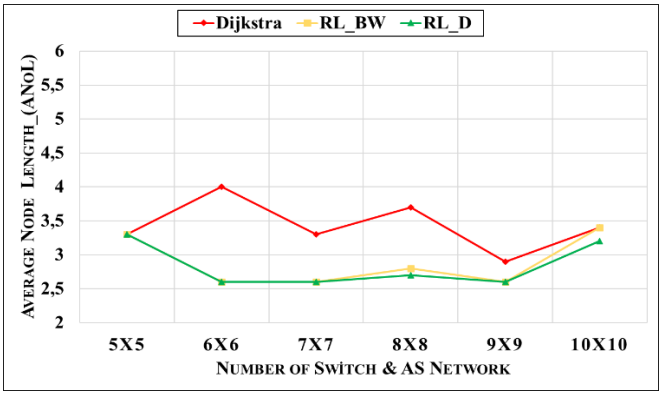
c) Same Request - Fixed-SW (10), Varying AS (5 - 10)



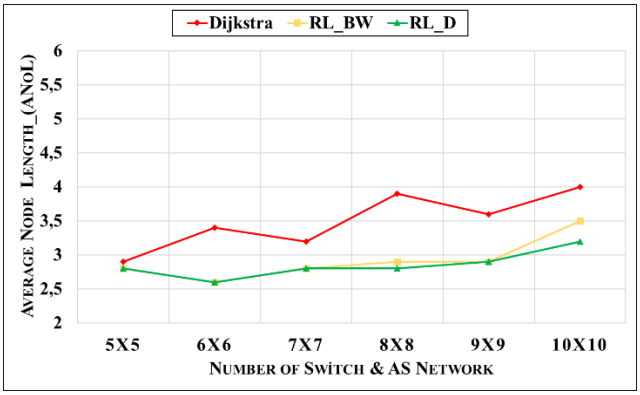
b) Different Request - Fixed-AS (10), Varying SW (5 - 10)



d) Different Request - Fixed-SW (10), Varying AS (5 - 10)



e) Same Request - Varying AS (5 - 10) , Varying SW (5 - 10)



f) Different Request - Varying AS (5 - 10) , Varying SW (5 - 10)

Figure 5.4 ANoL comparison of Dijkstra, RL_BW and RL_D

5.6 Average Network Length (ANL)

ANL metric quantifies the average number of ASes traversed along the E2E paths computed for a set of service demands. For each service demand SD_g.

Let N_g denote the collection of ASes that constitute the E2E path for SD_g, where $N_g \subseteq N$ and N is the set of all ASes configured in the simulation environment.

The network length for the service demand SD_g is then given by the cardinality $|N_g|$, which represents the number of ASes involved in the computed path.

Assuming that the routing algorithm has successfully computed E2E paths for R_s service demands, ANL is defined as:

ANL Equation:

$$\boldsymbol{ANL = \frac{1}{R_s} \sum_{g=1}^{R_s} |N_g|} \tag{14}$$

- N_g**:** For each service demand SD_g, N_g is the set of ASes forming the E2E path. Its cardinality $|N_g|$ represents the number of AS transitions encountered in that particular path.
- R_s**:** Denotes the total number of service demands for which a complete E2E path has been successfully determined.
- The numerator, $\sum_{g=1}^{R_s} |N_g|$, aggregates the total number of ASes traversed across all R_s successful paths.
- Dividing by R_s yields the mean number of ASes per successful routing path.

This metric provides an important measure of routing efficiency: a lower *ANL* indicates that, on average, the routing mechanism selects more direct paths (fewer AS boundaries

are crossed), whereas a higher *ANL* implies that the paths are longer, involving more inter-AS transitions.

ANL provides insight into the efficiency of different routing strategies by comparing Dijkstra, RL_BW, and RL_D under various network configurations. By examining how network traversal length evolves across different topological scenarios, this analysis highlights the adaptability of RL-based routing relative to conventional static path computation methods.

Figure 5.42a and Figure 5.43b analyze the effect of increasing switch counts on ANL while keeping the number of ASes constant. In the same request scenario (Figure 5.44a), Dijkstra-based routing demonstrates higher ANL values, fluctuating between 2.0 and 3.5, indicating longer network traversal due to its reliance on precomputed paths. As the switch count increases, RL_BW and RL_D maintain more efficient network traversal, stabilizing within the range of 2.0 to 2.9, showcasing their ability to dynamically adapt to network topology changes.

In the different request scenario (Figure 5.45b), Dijkstra's inefficiency becomes more apparent, reaching an ANL peak of 3.9 when the number of switches is low. Although it decreases as the switch count increases, it remains above 2.1, whereas RL-based models maintain significantly lower ANL values, consistently between 1.9 and 3.3. This confirms the advantage of RL-based dynamic routing in optimizing network traversal, particularly when network traffic patterns are not predetermined.

Figure 5.46c and Figure 5.47d evaluate the effect of increasing AS numbers while keeping the switch count fixed. In the same request scenario (Figure 5.48c), Dijkstra's ANL decreases from 3.0 to 2.1 as additional AS connections provide more routing flexibility. However, its reliance on static routing causes inefficiencies compared to RL-based models. RL_BW and RL_D maintain significantly lower ANL values, ranging from 2.0 to 2.2, ensuring more efficient network traversal.

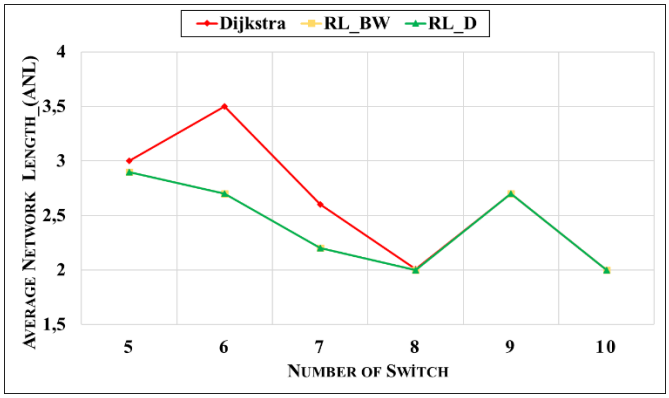
In the different request scenario (Figure 5.49d), the gap between static and adaptive routing becomes more evident. Dijkstra's ANL fluctuates between 1.9 and 3.9, whereas RL_BW and RL_D demonstrate more efficient network utilization, stabilizing between 1.8 and 2.9. These findings emphasize that RL-based routing is more effective in handling dynamic AS environments, where static routing struggles to adjust to topology variations.

Figure 5.50e and Figure 5.51f assess how ANL scales when both AS and switch counts vary. In the same request scenario (Figure 5.52e), Dijkstra's ANL fluctuates significantly, reaching 2.6 at its peak, while RL_BW and RL_D maintain consistently lower values, ranging from 2.0 to 2.6. RL-based models efficiently adapt to changes in network topology, ensuring minimal traversal length.

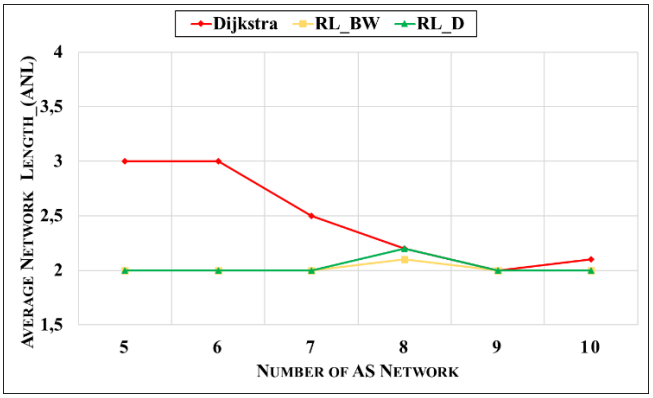
In the different request scenario (Figure 5.53f), Dijkstra exhibits increased variability, with ANL values ranging from 2.3 to 2.5, further emphasizing its limitations in adapting to large-scale network changes. In contrast, RL_BW and RL_D sustain lower traversal values between 2.0 and 2.2, reinforcing their efficiency in dynamically selecting optimal paths.

The ANL analysis in Figure 5.54 confirms that RL-based routing significantly outperforms Dijkstra in minimizing network traversal across different network configurations. Dijkstra’s static nature leads to inefficient path selection, resulting in higher ANL values, especially in dynamic network conditions where the topology is not precomputed.

Conversely, RL_BW and RL_D consistently reduce traversal length by adapting to real-time network states, ensuring shorter and more efficient paths. The findings suggest that RL-driven routing not only optimizes network traversal but also enhances overall network efficiency by reducing unnecessary hops, leading to improved QoS and better scalability in SDN environments.

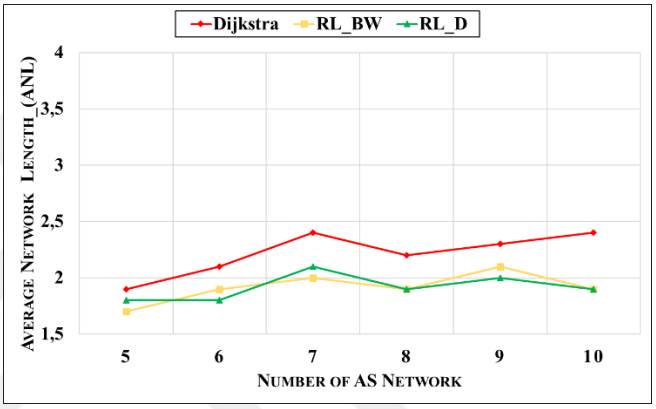


a) Same Request - Fixed-AS (10), Varying SW (5 - 10)

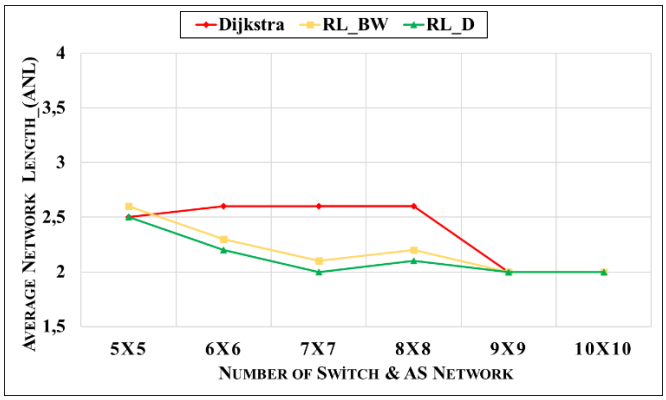


c) Same Request - Fixed-SW (10), Varying AS (5 - 10)

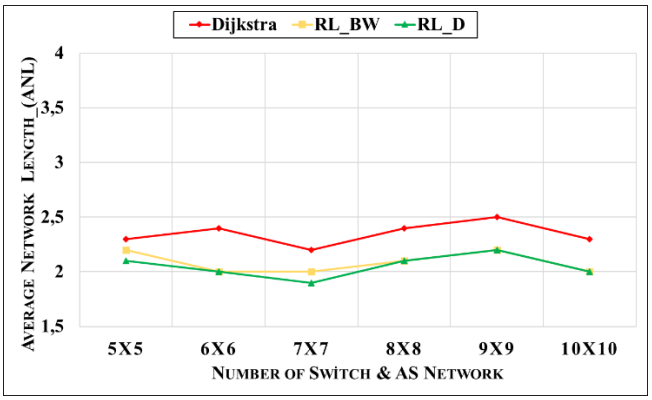
b) Different Request - Fixed-AS (10), Varying SW (5 - 10)



d) Different Request - Fixed-SW (10), Varying AS (5 - 10)



e) Same Request - Varying AS (5 - 10) , Varying SW (5 - 10)



f) Different Request - Varying AS (5 - 10) , Varying SW (5 - 10)

Figure 5.5 ANL comparison of Dijkstra, RL_BW and RL_D

6. DISCUSSION

The proposed framework, integrating DAG-based DLT with RL, demonstrates significant potential in overcoming the limitations of existing traffic management approaches in SDNs. While traditional routing methods rely on static policies, this framework enables a decentralized, scalable, and adaptive decision-making process through DAG-based validation. Additionally, by incorporating RL, routing decisions are dynamically optimized based on real-time network conditions, enhancing QoS compliance.

From a networking perspective, the framework offers substantial advantages, particularly in SDN environments where traditional routing algorithms, such as Dijkstra and OSPF, struggle to adapt to dynamic traffic patterns. Although widely used, conventional approaches often lack the ability to optimize paths dynamically according to network conditions. In contrast, the proposed model decentralizes routing decisions and utilizes DAG for high-efficiency transaction validation, reducing congestion, minimizing control overhead, and improving E2E latency. These characteristics make it particularly suitable for large-scale, multi-domain networks that require real-time responsiveness and adaptability.

The comparative evaluation against PRM and DRM further underscores the robustness of the framework. PRM, which relies on a centralized broker AS, provides structured traffic management but risks becoming a bottleneck under heavy network loads. On the other hand, DRM decentralizes routing decisions, increasing network resilience but potentially reducing efficiency due to coordination challenges. Additionally, the framework was evaluated against BCQRM to examine the differences between Blockchain and DAG implementations in SDN environments. While Blockchain provides strong security and data immutability, its sequential block processing results in relatively higher latency compared to DAG. However, simulation results indicate that the difference in latency is not significant, with DAG offering more advantages in terms of transaction validation scalability and parallel processing rather than pure speed improvements.

Beyond its impact on network efficiency, the framework also offers notable industrial and economic benefits. Its ability to handle high-throughput data with minimal latency makes it highly relevant for industries that rely on real-time communication, such as telecommunications, smart cities, and autonomous systems. Furthermore, by replacing Blockchain's energy-intensive consensus mechanisms with DAG-based validation, the framework improves energy efficiency and optimizes resource consumption. This approach aligns with sustainability goals by reducing the energy footprint of large-scale network infrastructures.

Despite these advantages, several challenges must be addressed to ensure the framework's practical applicability. One of the key concerns is the computational overhead associated with training RL models in dynamic and large-scale networks. RL-based decision-making requires significant processing power, posing limitations for real-time applications. Additionally, integrating DAG-based validation into existing SDN infrastructures presents challenges, as traditional network protocols are not inherently designed to support distributed ledger-based decision-making. Ensuring seamless interoperability between DAG-based routing updates and conventional networking protocols is crucial for real-world deployment.

Moreover, the scalability and transaction costs of Blockchain-based systems remain important limitations compared to DAG. Due to its block structure, Blockchain requires a set time interval and energy-intensive consensus mechanisms for transaction validation. Particularly, PoW consensus mechanisms increase both processing time and energy consumption, making Blockchain less efficient for high-throughput SDN networks. While DAG offers parallel transaction validation, reducing these limitations, it requires additional security measures to ensure data consistency and prevent malicious activities.

Additionally, the performance evaluation in this study was conducted using five key metrics: FET, MEE, ANL, ANoL, and APL. These metrics provide a comprehensive assessment of routing efficiency, message overhead, and network scalability. While these indicators effectively demonstrate the advantages of the proposed framework, future research can explore additional performance metrics such as packet loss rate and network

utilization to gain deeper insights into the impact of DAG-based RL routing on SDN environments.

Furthermore, maintaining transaction continuity and ensuring data integrity in DLT-based systems presents an ongoing challenge that requires balancing between Blockchain and DAG approaches. In Blockchain, historical transactions are immutable, whereas in DAG-based systems, transactions are continuously reinforced by new validations, which could introduce potential security vulnerabilities. Strengthening DAG's validation mechanisms and mitigating the risks posed by malicious nodes is crucial for ensuring system reliability.

Lastly, the integration of decentralized routing processes into traditional network management systems presents an operational challenge for network administrators. Conventional network management relies on centralized control mechanisms, whereas DAG and RL introduce distributed decision-making models that may require administrators to adapt to new monitoring and control paradigms. Addressing this challenge necessitates the development of new management tools and protocols tailored to support decentralized SDN traffic management.

The proposed DAG-based RL framework offers clear advantages over traditional and Blockchain-based routing methods. However, enhancing computational efficiency, strengthening security mechanisms in DAG-based architectures, and ensuring seamless integration with existing network management systems are critical for widespread adoption. Future research should focus on optimizing RL models for real-time adaptability, improving the integration of DAG-based validation within existing SDN infrastructures, and mitigating security vulnerabilities inherent to DAG systems. Additionally, ongoing efforts to reduce Blockchain transaction costs and improve energy efficiency will further contribute to the evolution of decentralized routing solutions.

In conclusion, the proposed framework successfully integrates RL and DAG-based DLT to optimize QoS-aware routing in SDN environments. While challenges remain in terms of computational efficiency, security, and interoperability with existing network

architectures, addressing these limitations will pave the way for a highly scalable, adaptive, and efficient traffic management solution for next-generation networks.

7. CONCLUSION

This thesis presented a comprehensive investigation into the integration of SDN, DLT, and RL to develop an adaptive and scalable traffic management framework. By addressing the limitations of existing routing mechanisms, this study introduced DQRL as a novel approach to optimizing traffic flow in modern networks. The proposed framework combines DAG-based DLT, which eliminates the need for block-based validation, with RL-driven adaptive routing, ensuring efficient and dynamic decision-making while maintaining QoS compliance.

Unlike traditional routing algorithms such as Dijkstra's shortest path algorithm and OSPF, which lack real-time adaptability, DQRL dynamically adjusts routing paths based on changing network conditions and QoS demands. The framework consists of two variations: DQRL-D, which prioritizes low-latency routing for delay-sensitive applications, and DQRL-BW, which optimizes throughput for bandwidth-intensive traffic flows. By leveraging Q-learning, the RL agent continuously improves its routing decisions, selecting the most efficient paths based on bandwidth availability, delay constraints, and reliability metrics.

To evaluate its performance, DQRL was benchmarked against three alternative frameworks: DRM, PRM, and BCQRM. DRM employs a decentralized routing model but lacks adaptive learning, leading to suboptimal path selection in dynamic environments. PRM enhances efficiency through a hierarchical structure but suffers from scalability limitations and increased control overhead. BCQRM, which integrates blockchain-based validation with QoS-aware routing, provides enhanced security and consistency, yet its block-based consensus mechanism introduces validation delays and computational overhead. DQRL outperforms all these approaches by integrating RL-driven adaptive learning with the high-speed validation capabilities of DAG-based transactions, achieving superior efficiency in QoS-aware traffic engineering.

Simulations were conducted using Mininet for network emulation, Python for RL implementation, and MATLAB for performance analysis. The network topology

consisted of 5 to 10 AS, each containing 5 to 10 nodes, ensuring a realistic and scalable test environment. Each experiment processed 20 traffic requests per iteration, allowing for statistical robustness and comprehensive performance evaluation. Key performance indicators included FET, MEE, APL, ANoL, and ANL.

The results demonstrated that DQRL consistently outperforms DRM, PRM, and BCQRM in multiple scenarios. Specifically, DQRL reduces FET by an average of 29%, with a maximum improvement of 42% over PRM in dynamic inter-domain environments. Even compared to BCQRM, the best-performing alternative, DQRL achieves 5% faster FET, highlighting the efficiency of its DAG-based transaction validation mechanism. Furthermore, DQRL reduces MEE by an average of 30%, significantly lowering the control overhead required for routing operations. Compared to BCQRM, DQRL achieves a 23% reduction in message exchange overhead, reinforcing the advantages of DAG's parallel transaction validation over traditional block-based architectures in managing high-throughput, real-time traffic environments.

The findings of this study establish DQRL as a highly efficient and scalable solution for next-generation networks, where real-time responsiveness, decentralized control, and QoS-aware traffic engineering are critical. The integration of DAG-based validation and RL not only surpasses conventional Dijkstra-based and blockchain-enhanced routing methods but also lays the foundation for further advancements in decentralized and autonomous traffic management systems.

Future research could explore alternative RL approaches to further enhance adaptability and learning efficiency in dynamic network environments. Additionally, advancements in DAG architectures and hybrid solutions integrating multiple ledger technologies could be investigated to improve scalability, security, and decentralized decision-making. Real-world deployment studies and integration with emerging network paradigms, such as 6G, edge computing, and Industrial IoT, would provide valuable insights into the practical feasibility and effectiveness of the proposed framework. By addressing these areas, this work lays a foundation for the continued evolution of intelligent, high-performance, and

adaptive traffic management systems in next-generation distributed networks, ensuring that future networking paradigms remain scalable, autonomous, and QoS-driven.

In conclusion, this research contributes to the fields of SDN, DLT, and RL by presenting a scalable, decentralized, and QoS-driven traffic management framework that addresses the increasing complexity of modern networks. By combining DAG-based transaction validation with RL-driven routing, DQRL optimizes network efficiency while providing an adaptive and intelligent foundation for future high-performance networking infrastructures. As networks continue to evolve, the principles established in this study pave the way for more autonomous, efficient, and QoS-aware traffic management solutions in next-generation distributed architectures.

REFERENCES

Abdulqadder, I. H., & Zhou, S. (2022). SliceBlock: Context-Aware Authentication Handover and Secure Network Slicing Using DAG-Blockchain in Edge-Assisted SDN/NFV-6G Environment. IEEE Internet of Things Journal, 9, 18079-18097.

Agarwal, S., Bailey, C., & Benson, T. (2013). "Software Defined Networking and its Application in Large Data Centers." IEEE Communications Magazine, 51(2), 12-19. https://doi.org/10.1109/MCOM.2013.6476866

Agarwal, S., Kodialam, M., & Lakshman, T. V. (2013). Traffic engineering in software-defined networks. Proceedings IEEE INFOCOM, 2211-2219.

Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015). Security in Software Defined Networks: A Survey. IEEE Communications Surveys & Tutorials, 17, 2317-2346.

Akhunzada, A., Gani, A., Anuar, N. B., Abdelaziz, A., Khan, M., Hayat, A., & Khan, S. (2016). Secure and dependable software-defined networks. Journal of Network and Computer Applications, 61, 199-221.

Akhunzada, A., Gani, A., Khan, S., & Anuar, N. B. (2016). "Data Layer in software-defined networking: Comparative study of data Layer implementations." Journal of Network and Computer Applications, 70, 32-42. https://doi.org/10.1016/j.jnca.2016.02.011

Allison, C. (1998). Quality of service for wide area clusters. Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications.

Alotaibi, J., & Alazzawi, L. (2021). Safiov: A secure and fast communication in fog-based internet-of-vehicles using SDN and blockchain. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)* (pp. 334–339).

Alsaeedi, M., Mohamad, M. M., & Al-Roubaiey, A. A. (2019). Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey. IEEE Access, 7, 107346-107379.

Apostolopoulos, G., Kamat, S., Williams, D., Guérin, R., Orda, A., & Przygienda, T. (1999). QoS Routing Mechanisms and OSPF Extensions. RFC 2676, 1-50.

Arora, P. (2018). The Heart and Brain of SDN: SDN Controllers. In Software-Defined Networking.

Ashwini, J., Sushma, M., & Sanjay, H. A. (2011). Queuing delay aware path selection algorithm as extension to OSPF. 3rd International Conference on Electronics Computer Technology, 99-103.

Asseman, A., Antoine, N., & Ozcan, A. S. (2021). Accelerating Deep Neuroevolution on Distributed FPGAs for Reinforcement Learning Problems. ACM Journal on Emerging Technologies in Computing Systems (JETC), 17, 1–17.

Ayoubi, S., Kim, K., Mehaoua, A., & Shin, M. (2018). "Machine learning for network performance prediction and QoS in SDN-enabled networks." IEEE Network, 32(6), 168-175. https://doi.org/10.1109/MNET.2018.1800178

Azevedo, T., Silva, P., & Cruz, J. (2021). Adaptive Traffic Signal Control with Q-Learning and DQN. Journal of Intelligent Transportation, 17(2), 123-145.

Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). Performance Evaluation of a Scalable Software-Defined Networking Deployment. Second European Workshop on Software Defined Networks, 68-74.

Bassene, A., & Gueye, B. (2022). A Self-adaptive QoS-management Framework for Highly Dynamic IoT Networks. IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD), 1-8.

Benčić, F. M., & Žarko, I. P. (2018). "Distributed ledger technology: Blockchain compared to directed acyclic graph." Journal of Industrial Engineering and Management, 11(4), 511-534. https://doi.org/10.3926/jiem.2711

Benčić, F. M., & Žarko, I. P. (2018). "Distributed ledger technology: Blockchain compared to directed acyclic graph." Journal of Industrial Engineering and Management, 11(4), 511-534. https://doi.org/10.3926/jiem.2711

Benčić, F. M., & Žarko, I. P. (2018). Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 1569-1570.

Bikos, A. N., & Kumar, S. A. P. (2021). Reinforcement Learning-Based Anomaly Detection for Internet of Things Distributed Ledger Technology. 2021 IEEE Symposium on Computers and Communications (ISCC), 1-7.

Bilal, M., & Khan, M. (2019). "The data and control Layer separation in SDN: A survey." Computer Networks, 143, 251-266. https://doi.org/10.1016/j.comnet.2018.12.002

Bilal, R., & Khan, B. (2019). Software-Defined Networks (SDN). Handbook of Research on Cloud Computing and Big Data Applications in IoT.

Birkner, R., Canini, M., Feldmann, A., & Schmid, S. (2016). "OpenFlow-based control Layer latency-aware and fault-tolerant controller placement." Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies, 15, 1-13. https://doi.org/10.1145/2999572.2999600

Cao, Y., Ren, X., Qiu, C., & Wang, X. (2022). Hierarchical reinforcement learning for blockchain-assisted software defined industrial energy market. *IEEE Transactions on Industrial Informatics*, *18*(9), 6100–6108.

Carvalho, R. N., Costa, L., Bordim, J., & Alchieri, E. (2020). New Programmable Data Layer Architecture Based on P4 OpenFlow Agent.

Casas-Velasco, D. M., Rendón, O., & da Fonseca, N. D. (2021). Intelligent Routing Based on Reinforcement Learning for Software-Defined Networking. IEEE Transactions on Network and Service Management, 18, 870–881.

Casino, F., Dasaklis, T. K., & Patsakis, C. (2019). "A systematic literature review of blockchain-based applications: Current status, classification and open issues." Telematics and Informatics, 36, 55-81. https://doi.org/10.1016/j.tele.2018.11.006

CBT Nuggets. (2020, April 15). What is QoS in networking. Retrieved from https://www.cbtnuggets.com/blog/technology/networking/what-is-qos-in-networking

Chen, C., Xue, F., Lu, Z., Tang, Z., & Li, C. (2022). RLMR: Reinforcement learning based multipath routing for SDN. *Wireless Communications and Mobile Computing, 2022*(1), Article 5124960.

Chen, C., Chen, X., & Fang, Z. (2022). TIPS: Transaction Inclusion Protocol With Signaling in DAG-Based Blockchain. IEEE Journal on Selected Areas in Communications, 40, 3685-3701.

Chen, S., & Cobb, J. (2006). Wireless Quality-of-Service support. Wireless Networks, 12, 409-410.

Chen, Y.-R., Rezapour, A., Tzeng, W.-G., & Tsai, S.-C. (2020). RL-Routing: An SDN Routing Algorithm Based on Deep Reinforcement Learning. IEEE Transactions on Network Science and Engineering, 7, 3185-3199.

Chica, J. C., Imbachi, J. C., & Botero, J. F. (2020). Security in SDN: A comprehensive survey. Journal of Network and Computer Applications, 159, 102595.

Chien, A., & Kim, J. H. (1997). Approaches to Quality of Service in High-Performance Networks. Springer.

Chiu, K.-C., Liu, C.-C., Chou, L.-D., et al. (2022). Reinforcement learning-based service-oriented dynamic multipath routing in SDN. Wireless Communications and Mobile Computing, 2022.

Chu, T., Wang, J., Codecà, L., & Li, Z. (2019). Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems, 21, 1086-1095.

Chun, M. (2003). Implementation Algorithm of OSPF Protocol. Journal of Hangzhou Institute of Electronic Engineering.

Cisco. (2018). *Cisco Visual Networking Index (VNI) complete forecast update, 2017–2022: Global presentation*. Retrieved from https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html

Conti, M., Kumar, S., Lal, C., & Ruj, S. (2018). "A survey on security and privacy issues of blockchain technology." IEEE Communications Surveys & Tutorials, 21(2), 1059-1079. https://doi.org/10.1109/COMST.2018.2863957

Coskun, M., Baggag, A., & Chawla, S. (2018). Deep Reinforcement Learning for Traffic Light Optimization. 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 564-571.

Cullen, A., Ferraro, P., King, C., & Shorten, R. (2020). On the Resilience of DAG-Based Distributed Ledgers in IoT Applications. IEEE Internet of Things Journal, 7, 7112-7122.

Cullen, A., Ferraro, P., King, C., & Shorten, R. (2020). On the Resilience of DAG-Based Distributed Ledgers in IoT Applications. IEEE Internet of Things Journal, 7, 7112-7122.

Cullen, A., Perez, D., & Vukolic, M. (2020). "The security of DAG-based distributed ledgers in the presence of selfish mining attacks." IEEE Transactions on

Information Forensics and Security, 15, 1-10. https://doi.org/10.1109/TIFS.2020.2975584

Datacenter Systems. (2024). Role of Software-Defined Networking (SDN) in Data Center Connectivity. Retrieved from Datacenter Systems.

Desai, R., & Patil, B. (2017). Adaptive routing based on predictive reinforcement learning. International Journal of Computers and Applications, 40, 73–81.

Dixit, V., Kyung, S., Zhao, Z., Doupé, A., Shoshitaishvili, Y., & Ahn, G. J. (2018). Challenges and Preparedness of SDN-based Firewalls. Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization.

Dong, Z., Zheng, E., Lee, Y. C., & Zomaya, A. Y. (2019). DAGBENCH: A Performance Evaluation Framework for DAG Distributed Ledgers. 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 264-271.

Ejable. (2023). Types of machine learning: AI, machine learning, and deep learning. Retrieved from https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/types-of-machine-learning/

Elzain, H., & Yang, W. (2018). QoS-aware Topology Discovery in Decentralized Software Defined Wireless Mesh Network (D-SDWMN) Architecture. International Conference Proceedings.

Faizullah, M., & Almutairi, M. (2018). "Efficient data forwarding with reduced processing time in SDN data Layer." IEEE Transactions on Network and Service Management, 15(4), 17-24. https://doi.org/10.1109/TNSM.2018.2879455

Faizullah, S., & Almutairi, S. (2018). Vulnerabilities in SDN Due to Separation of Data and Control Layers. International Journal of Computer Applications, 179, 21-24.

Fan, C. (2019). Performance Analysis and Design of an IoT-Friendly DAG-based Distributed Ledger System.

Fan, C., Ghaemi, S., Khazaei, H., Chen, Y., & Musílek, P. (2021). Performance Analysis of the IOTA DAG-Based Distributed Ledger. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 6, 10:1-10:20.

Fan, L., Zhang, R., Liu, Y., & Song, H. (2021). "Blockchain or DAG for IoT applications Analyzing energy efficiency and performance of blockchain and DAG in IoT scenarios." IEEE Internet of Things Journal, 8(4), 3214-3225. https://doi.org/10.1109/JIOT.2020.3033066

Ferraro, P., Shorten, R., & King, C. (2020). On the Stability of Unverified Transactions in a DAG-Based Distributed Ledger. IEEE Transactions on Automatic Control, 65, 3772-3783.

Fiveable. (2024). Benefits and challenges of SDN adoption. Retrieved from Fiveable Library.

Fiveable. (2024). Key Components of SDN Architecture. Retrieved from Fiveable Library.

Fiveable. (2024). SDN architecture: Control and Data Layer Separation. Retrieved from Fiveable Library.

Gandhi, D., & Sajnani, A. (2020). Network Quality of Service. Springer, 101-106.

GeeksforGeeks. (n.d.). Software Defined Networking (SDN). Retrieved from https://www.geeksforgeeks.org/software-defined-networking/

Ghalwash, H., & Huang, C.-H. (2020). A congestion control mechanism for SDN-based fat-tree networks. 2020 IEEE High Performance Extreme Computing Conference (HPEC), 1-7.

Gray, N., Zinner, T., Gebert, S., & Tran-Gia, P. (2016). Simulation Framework for Distributed SDN-Controller Architectures in OMNeT++.

Guler, E., Karakus, M., & Uludag, S. (2022). *SpectrumChain: An efficient spectrum management framework in blockchain-enabled flexible SDONs*. In ICC 2022-IEEE International Conference on Communications (pp. 5744–5749). IEEE. https://doi.org/10.1109/ICC45855.2022.9838747

Guerzoni, R., Hecker, A., Karagiannis, G., & Markopoulou, M. (2014). "Network function virtualization: Challenges and solutions for security and quality of service." Computer Networks, 73, 220-229. https://doi.org/10.1016/j.comnet.2014.08.014

Guo, X., Lin, H., Li, Z., & Peng, M. (2020). Deep-reinforcement-learning-based QoS-aware secure routing for SDN-IoT. *IEEE Internet of Things Journal*, *7*(7), 6242–6251.

He, K., Khalid, J., Gember, A., Das, S., Prakash, C., Akella, A., Li, E. L., & Thottan, M. (2015). Measuring Control Layer Latency in SDN-enabled Switches. Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research.

Hedera. (n.d.). DAG vs Blockchain. Retrieved from https://hedera.com/learning/distributed-ledger-technologies/dag-vs-blockchain.

Hegazy, A., & El-Aasser, M. (2021). Network Security Challenges and Countermeasures in SDN Environments. 2021 Eighth International Conference on Software Defined Systems (SDS), 1-8.

Heller, B., Sherwood, R., & McKeown, N. (2012). "The controller placement problem." Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 7-12. https://doi.org/10.1145/2342441.2342444

Hossain, M. (2020). QoS-Aware Traffic Management in Software Defined Networks: A Machine Learning Perspective. Master's Thesis, University of Technology.

Hua, S., & Qu, G. (2003). A new quality of service metric for hard/soft real-time applications. Proceedings ITCC 2003, 347-351.

IANA (Internet Assigned Numbers Authority). (2020, May 31). A Quick Look at IANA Services. Retrieved from https://www.iana.org/

Ibrar, M., Wang, L., Muntean, G.-M., Chen, J., Shah, N., & Akbar, A. (2021). IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems. *IEEE Internet of Things Journal*, *8*(5), 3130–3142.

ICODA. (2021, February 5). A Quick Look at DAG vs. Blockchain Technology. Retrieved from https://icoda.io/blog/dag-vs-blockchain/.

IEEE Xplore. (2024). On the (in)Security of the Control Layer of SDN Architecture: A Survey. IEEE Journals & Magazine.

Islam, M. T., Islam, N., & Al Refat, M. (2020). Node to Node Performance Evaluation through RYU SDN Controller. Wireless Personal Communications, 112, 555-570.

Jadhao, N. S., & Jadhao, A. S. (2014). Traffic Signal Control Using Reinforcement Learning. 2014 Fourth International Conference on Communication Systems and Network Technologies, 1130-1135.

Jain, R., & Paul, S. (2013). "Network virtualization and software defined networking for cloud computing: A survey." IEEE Communications Magazine, 51(11), 24-31. https://doi.org/10.1109/MCOM.2013.6658646

Javatpoint. (2023). Software Defined Networking (SDN): Benefits and Challenges of Network Virtualization. Retrieved from Javatpoint.

Karakus, M., & Durresi, A. (2017). QoS in Software Defined Networking (SDN): A survey. J. Netw. Comput. Appl., 80, 200-218.

Karakus, M., Güler, E., & Uludag, S. (2021). QoSChain: Provisioning inter-AS QoS in software-defined networks with blockchain. IEEE Transactions on Network and Service Management, 18(2), 1706–1717.

Karakus, M., & Guler, E. (2020). “Routingchain: A proof-of-concept model for a blockchain-enabled qos-based inter-as routing in sdn,” in 2020 IEEE International Black Sea Conference on Communications and Networking BlackSeaCom), 2020, pp. 1–6.

Karakus, M. (2023). Implementation of Blockchain-Assisted Source Routing for Traffic Management in Software-Defined Networks. Duzce University Journal of Science and Technology, 11(3), 1250-1268.

Karakuş, M., Güler, E., & Uludağ, S. (2023). SmartContractChain (SC2): Cross-ISP QoS traffic management framework with SDN and blockchain. *Peer-to-Peer Networking and Applications, 16*, 3003–3020.

Katta, N. P., Zhang, J., & Rexford, J. (2016). "Ravel: A database-defined network." Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 33-39. https://doi.org/10.1145/3005745.3005758

Ke, C.-H., Tu, Y.-H., & Ma, Y.-W. (2023). A reinforcement learning approach for widest path routing in software-defined networks. *ICT Express, 9*(5), 882–889.

Kelly, S., & Heywood, M. (2018). Emergent Solutions to High-Dimensional Multitask Reinforcement Learning. Evolutionary Computation, 26, 347–380.

Khan, M., Hartog, F., & Hu, J. (2024). Toward Verification of DAG-Based Distributed Ledger Technologies through Discrete-Event Simulation. Sensors, 24(5), 1583.

Kim, G., Kim, Y., & Lim, H. (2022). Deep reinforcement learning-based routing on software-defined networks. *IEEE Access, 10*, 18121–18133.

Kota, S., Pahlavan, K., & Leppanen, P. (2004). Quality of Service in IP Networks. In Networking (pp. 255-271). Springer.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). "Software-defined networking: A comprehensive survey." Proceedings of the IEEE, 103(1), 14-76. https://doi.org/10.1109/JPROC.2014.2371999

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). "Software-defined networking: A comprehensive survey." Proceedings of the IEEE, 103(1), 14-76. https://doi.org/10.1109/JPROC.2014.2371999

Lee, G., Lee, C., & Roh, B. (2022). QoS Support Path Selection for Inter-Domain Flows Using Effective Delay and Directed Acyclic Graph in Multi-Domain SDN. Electronics.

Lee, J., Chung, J., & Sohn, K. (2020). Reinforcement Learning for Joint Control of Traffic Signals in a Transportation Network. IEEE Transactions on Vehicular Technology, 69, 1375-1387.

Li, Y., & Chen, M. (2017). "Software-defined network function virtualization: A survey." IEEE Access, 5, 2542-2563. https://doi.org/10.1109/ACCESS.2017.2676098

Li, Y., Zhang, D., Taheri, J., & Li, K. (2018). SDN Components and OpenFlow. OpenFlow Protocol.

Lin, Y., Qu, G., Huang, L., & Wierman, A. (2020). Distributed Reinforcement Learning in Multi-Agent Networked Systems. ArXiv.

Lin, H., Garg, S., Hu, J., Kaddoum, G., Peng, M., & Hossain, M. S. (2021). Blockchain and deep reinforcement learning empowered spatial crowdsourcing in software-defined internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems, 22*(6), 3755–3764.

Lu, X., & Jiang, C. (2023). TEEDAG: A High-Throughput Distributed Ledger Based on TEE and Directed Acyclic Graph. Electronics.

Luo, J., Chen, Q., Yu, F. R., & Tang, L. (2020). Blockchain-enabled software-defined industrial internet of things with deep reinforcement learning. *IEEE Internet of Things Journal*, *7*(6), 5466–5480.

Luo, J., Yu, F. R., Chen, Q., & Tang, L. (2020). Blockchain-enabled software-defined industrial internet of things with deep recurrent Q-network. In *ICC 2020 – 2020 IEEE International Conference on Communications (ICC)* (pp. 1–6).

MarketsandMarkets. (2023, November). *Software-defined networking market global forecast to 2028 (USD BN)*. Retrieved [Erişim Tarihi], from https://www.marketsandmarkets.com/Market-Reports/software-defined-networking-sdn-market-655.html

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). "OpenFlow: Enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review, 38(2), 69-74. https://doi.org/10.1145/1355734.1355746

Mellouk, A., Hoceini, S., & Amirat, Y. (2007). Adaptive quality of service-based routing approaches: Development of neuro-dynamic state-dependent reinforcement learning algorithms. International Journal of Communication Systems, 20, 1113–1130.

Mendiola, A., Jacob, E., Aguado, A., & Zabala, A. (2017). "A survey on the contributions of software-defined networking to traffic engineering." IEEE Communications Surveys & Tutorials, 19(2), 918-953. https://doi.org/10.1109/COMST.2017.2654695

Meneguette, R. (2020). Software-Defined Networks: Challenges for SDN as an Infrastructure for Intelligent Transport Systems based on Vehicular Networks. 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 205-212.

Meneguette, R., Fonseca, D., & Napoli, A. (2020). "QoS-aware SDN data Layer solutions." Future Generation Computer Systems, 110, 614-628. https://doi.org/10.1016/j.future.2020.04.030

Miller, R., Ahmed, S., & Patel, K. (2022). Applications and Challenges of DAG Structures in Distributed Ledgers. Journal of Advanced Distributed Technologies, 20(4), 345-359.

Mönch, C., & Rizk, A. (2023). Directed Acyclic Graph-Type Distributed Ledgers via Young-Age Preferential Attachment. Stochastic Systems.

Mönch, M., & Rizk, R. (2023). "DAG-based distributed ledgers and their implications for blockchain scalability." Journal of Emerging Technologies in Computing Systems, 19(2), 121-130. https://doi.org/10.1145/3488993

Mushtaq, A., Haq, I., Sarwar, M. A., Khan, A., Khalil, W., & Mughal, M. A. (2023). Multi-Agent Reinforcement Learning for Traffic Flow Management of Autonomous Vehicles. Sensors, 23(5).

Nakamoto, S. (2008). "Bitcoin: A peer-to-peer electronic cash system." Bitcoin.org. Retrieved from https://bitcoin.org/bitcoin.pdf

Next Horizon. (2024). Benefits of Software-Defined Networking. Retrieved from Next Horizon.

Nguyen, T.-T., Bonnet, C., & Härri, J. (2016). SDN-based distributed mobility management for 5G networks. IEEE Wireless Communications and Networking Conference.

Niu, Z., Shu, T., & Takahashi, Y. (2003). A vacation queue with setup and close-down times and batch Markovian arrival processes. Performance Evaluation, 54, 225-248.

Oliveira, A. T., Martins, B., Moreno, M., Vieira, A., Gomes, A. A., & Ziviani, A. (2018). SDN-Based Architecture for Providing QoS to High-Performance Distributed Applications. IEEE Symposium on Computers and Communications, 602-607.

Open Networking Foundation. (2024). Software-Defined Networking: The New Norm for Networks.

Park, S., & Kim, H. (2019). DAG-Based Distributed Ledger for Low-Latency Smart Grid Network. Energies.

Peng, S., Liu, Y., Chen, J., He, J., & Wang, Y. (2023). Komorebi: A DAG-based Asynchronous BFT Consensus via Sharding. 2023 IEEE Symposium on Computers and Communications (ISCC), 1221-1227.

Pervez, H., Muneeb, M., Irfan, M. U., & Haq, I. (2018). A Comparative Analysis of DAG-Based Blockchain Architectures. 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), 27-34.

Pham, Q. T. A., Hadjadj-Aoul, Y., & Outtagarts, A. (2018). Deep reinforcement learning based QoS-aware routing in knowledge-defined networking. In *Proceedings of the 14th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness* (pp. 1–13). Ho Chi Minh City, Vietnam.

Popov, S. (2017). "The Tangle." IOTA Foundation. Retrieved from https://iota.org/IOTA_Whitepaper.pdf

Prostov, I. A., Amfiteatrova, S. S., & Butakova, N. G. (2021). Construction and Security Analysis of Private Directed Acyclic Graph Based Systems for Internet of Things. 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2394-2398.

Prostov, Y., Kudryavtsev, S., & Zavyalov, Y. (2021). "Fault tolerance and scalability in DAG-based distributed ledger technology." International Journal of Distributed Systems and Technologies, 12(3), 45-56. https://doi.org/10.4018/IJDST

Qi, D., & Shen, S. (2017). CFCC: A Controller Framework Supporting Component-Based SDN Applications Development. Future Generation Computer Systems, 10, 73-86.

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

Rouse, M. (2021, May 25). Distributed ledger technology (DLT). TechTarget. Retrieved from https://www.techtarget.com/searchcio/definition/distributed-ledger

Sallam, A., Refaey, A., & Shami, A. (2019). On the Security of SDN: A Completed Secure and Scalable Framework Using the Software-Defined Perimeter. IEEE Access, 7, 146577-146587.

Sankar, L. S., Sindhu, M., & Sethumadhavan, M. (2017). "Survey of consensus protocols on blockchain applications." Proceedings of the Fourth International Conference on Advanced Computing and Communication Systems, 1-5. https://doi.org/10.1109/ICACCS.2017.8014672

Sarwar, M. A., Khan, A., Khalil, W., & Mughal, M. A. (2023). Multi-Agent Reinforcement Learning for Traffic Flow Management. Sensors, 23(5), 2373.

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2015). "A survey of security in software-defined networks." IEEE Communications Surveys & Tutorials, 18(1), 623-654. https://doi.org/10.1109/COMST.2015.2417566

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2016). A Survey of Security in Software Defined Networks. IEEE Communications Surveys & Tutorials, 18, 623-654.

SDxCentral. (2020). Understanding the SDN Architecture and SDN Control Layer. Retrieved from SDxCentral.

Sewak, M. (2019). Temporal Difference Learning, SARSA, and Q-Learning. In Deep Reinforcement Learning.

Sharma, N. (2024). Software-Defined Networking | Enhancing Network Efficiency. Retrieved from Nitiz Sharma.

Shin, M. K., Nam, J., & Kim, H. (2014). "Software-defined networking (SDN): A reference architecture and open APIs." International Journal of Software Engineering and Its Applications, 8(4), 105-110.

Sokappadu, B., Hardin, A., Mungur, A., & Armoogum, S. (2019). Software Defined Networks: Issues and Challenges. 2019 Conference on Next Generation Computing Applications (NextComp), 1-5.

Song, Y., Qian, X., Zhang, N., Wang, W., & Xiong, A. (2023). QoS Routing Optimization Based on Deep Reinforcement Learning in SDN. Journal of Network and Systems Management, 31(4), 987–1005.

Stampa, G., Arias, M., Sanchez-Charles, D., Muntes-Mulero, V., & Cabellos, A. (2017). A deep-reinforcement learning approach for software-defined networking routing optimization. *arXiv preprint arXiv:1709.07080*.

Statista. (2024, September 24). *Size of distributed ledger market worldwide 2020-2030, by use case*. Statista. https://www.statista.com/statistics/1259858/distributed-ledger-market-size-use-case-worldwide/

Sufiev, H., Haddad, Y., & Barenboim, L. (2019). Dynamic SDN Controller Load Balancing. Future Internet, 11, 75.

Sun, P., Hu, Y., Lan, J., Tian, L., & Chen, M. (2019). TIDE: Time-relevant deep reinforcement learning for routing optimization. Future Generation Computer Systems, 99, 401–409.

Supermicro. (2024). What is Software-Defined Networking (SDN)?. Retrieved from Supermicro.

Wang, Q., Yu, J., Chen, S., & Xiang, Y. (2022). SoK: DAG-based Blockchain Systems. ACM Computing Surveys.

Wang, R., Liu, C., & Kim, J. (2021). DAG-Based DLT Framework for Cross-Network Traffic Management in SDNs. IEEE Transactions on Emerging Topics in Computing, 10(3), 549–561.

Wang, Y., et al. (2022). Deep Reinforcement Learning for Urban Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems, 23(8), 4567-4581.

Wang, Y.-H., Li, T.-H. S., & Lin, C.-J. (2013). Backward Q-learning: The combination of Sarsa algorithm and Q-learning. Eng. Appl. Artif. Intell., 26, 2184–2193.

Watanabe, Y., & Shoji, Y. (2023). An Application of DAG-based Distributed Ledger to Manage Content Whereabouts for Beyond 5G Networks. 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), 903-904.

Watanabe, Y., & Shoji, Y. (2023). An Application of DAG-based Distributed Ledger to Manage Content Whereabouts for Beyond 5G Networks. 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), 903-904.

WoolyPooly. (2023, August 23). DAG vs Blockchain: 10 Key Differences You Need to Know. Retrieved from https://woolypooly.com/en/blog/dag-vs-blockchain.

Worldstream. (2023). 8 Advantages of Using SDN for IT Infrastructure Deployment. Retrieved from Worldstream.

Wu, J., Li, K., & Jia, Q.-S. (2020). Decentralized Multi-agent Reinforcement Learning with Multi-time Scale of Decision Epochs. 2020 59th IEEE Conference on Decision and Control (CDC), 578-584.

Xiangyun, Z., Lijun, W., Zhiyuan, L., & Yulin, J. (2021). Deep Reinforcement Learning with Graph Convolutional Networks for Load Balancing in SDN-Based Data Center Networks. 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 344-352.

Xiao, Y., Zhang, N., Lou, W., & Hou, Y. T. (2020). "A survey of distributed consensus protocols for blockchain networks." IEEE Communications Surveys & Tutorials, 22(2), 1432-1465. https://doi.org/10.1109/COMST.2020.2966147

Xie, H., Huo, Q., Zhang, X., & Zhao, J. (2016). "Research on load balancing based on SDN." Procedia Computer Science, 83, 913-917. https://doi.org/10.1016/j.procs.2016.04.190

Xu, C., Chen, B.-L., & Qian, H.-Y. (2015). Quality of Service Guaranteed Resource Management Dynamically in Software Defined Network. Journal of Communications, 10, 843-850.

Xu, Z.-x., Cao, L., Chen, X.-l., Li, C., Zhang, Y., & Lai, J. (2018). Deep Reinforcement Learning with Sarsa and Q-Learning: A Hybrid Approach. IEICE Trans. Inf. Syst., 101-D, 2315–2322.

Yang, H., Ivey, J. S., & Riley, G. (2017). Scalability comparison of SDN control Layer architectures based on simulations. IEEE 36th International Performance Computing and Communications Conference (IPCCC), 1-8.

Yang, S., Cui, L., Chen, Z., & Xiao, W. (2020). An Efficient Approach to Robust SDN Controller Placement for Security. IEEE Transactions on Network and Service Management, 17, 1669-1682.

Yau, K., Qadir, J., Khoo, H. L., Ling, M. H., & Komisarczuk, P. (2017). A Survey on Reinforcement Learning Models and Algorithms for Traffic Signal Control. ACM Computing Surveys (CSUR), 50(1), 1-38.

Ye, M., Huang, L. Q., Wang, X. L., Wang, Y., Jiang, Q. X., & Qiu, H. B. (2024). A new intelligent cross-domain routing method in SDN based on a proposed multiagent reinforcement learning algorithm. *International Journal of Intelligent Computing and Cybernetics*.

Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). "Where is current research on blockchain technology?—A systematic review." PLOS ONE, 11(10), e0163477. https://doi.org/10.1371/journal.pone.0163477

Younus, M., Khan, M. K., Anjum, M., Afridi, S., Arain, Z. A., & Jamali, A. A. (2021). Optimizing the Lifetime of Software Defined Wireless Sensor Network via Reinforcement Learning. IEEE Access, 9, 259–272.

Yu, C., Lan, J., Guo, Z., & Hu, Y. (2018). DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. IEEE Access, 6, 64533–64543.

Zhang, T., & Li, X. (2022). Deep Reinforcement Learning-based QoS Aware Routing in Software Defined Networks. IEEE Transactions on Network and Service Management, 19(1), 35–48.

Zhang, Z., Zhu, D., & Mi, B. (2020). C-DAG: Community-Assisted DAG Mechanism with High Throughput and Eventual Consistency. Springer International Publishing.

Zhang J, Ye M, Guo Z, Yen CY, Chao HJ (2020) CFR-RL: Traffic engineering with reinforcement learning in SDN. IEEE J Sel Areas Commun 38(10):2249–2259

Zhao, N., Liang, Y.-C., Niyato, D., Pei, Y., Wu, M., & Jiang, Y. (2019). Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Cellular Networks. IEEE Transactions on Wireless Communications, 18, 5141–5152.

Zhao, X., & Wu, J. (2017). "An overview of SDN data Layer implementations." ACM Computing Surveys, 50(4), 54-66. https://doi.org/10.1145/3097336

Zhao, Z., & Wu, B. (2017). Scalable SDN architecture with distributed placement of controllers for WAN. Concurrency and Computation: Practice and Experience, 29.

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). "An overview of blockchain technology: Architecture, consensus, and future trends." Proceedings of the IEEE, 105(9), 1879-1896. https://doi.org/10.1109/JPROC.2017.2705918

APPENDIX 1 LIST OF EQUATIONS