Cross-Layer Scheduling and Routing for Ultra Reliable and Low Latency Communication

by

Xiangyu Ren
B.Eng., University of Electronic Science and Technology of China, 2019

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Cross-Layer Scheduling and Routing for Ultra Reliable and Low Latency Communication

by

Xiangyu Ren
B.Eng., University of Electronic Science and Technology of China, 2019

Supervisory Committee

Dr. Lin Cai. Supervisor Main, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Kin Fun Li, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Sudhakar Ganti, Outside Member
(Department of Computer Science)

Contents

Supervisory Committee **ii**

Table of Contents **iii**

List of Tables **viii**

List of Figures **ix**

Abstract **xiii**

Acknowledgements **xvi**

Dedication **xvii**

1 Introduction **1**

1.1 Background 1

1.2 Design Principle of New Protocols for Ultra-Reliable-and-Low-Latency Communications 3

1.3 Proposed Research Issues and Contributions 6

1.3.1 Congestion-aware delay-guaranteed scheduling and routing with renewal optimization 6

1.3.2 Downlink Scheduler for Delay Guaranteed Services Using Deep Reinforcement Learning 7

1.3.3 QoS-guaranteed Routing Protocol for Intelligent Transportation System: A Hybrid Approach 8

1.3.4 QMAC: Resource Allocation for Advanced Vehicular-to-Everything (V2X) Application 9

2 Congestion-aware Delay-guaranteed Scheduling and Routing with Renewal Optimization 10

2.1 Introduction 11
2.2 Related works 14
2.2.1 Delay-guaranteed Network Architecture 14
2.2.2 Delay-based Scheduling and Routing Protocol 14
2.3 System Model and Problem Formulation 15
2.3.1 Network Architecture 15
2.3.2 Network Model 20
2.3.3 Weighted-Round-Robin Scheduling Model 21
2.3.4 Queuing Model 22
2.3.5 Problem Formulation 22
2.4 Algorithm Design 24
2.4.1 Delay-guaranteed Scheduling and Routing Protocol 24
2.4.2 Network Utility Maximization 28
2.5 Performance evaluation 32
2.5.1 Network Configurations 33
2.5.2 Performance Metrics 33
2.5.3 Simulation results and analysis 34
2.6 Conclusion 42

3 Downlink Scheduler for Delay Guaranteed Services Using Deep Reinforcement Learning 43

3.1 Introduction 44
3.2 Related Works 46
3.3 System Model and Problem Formulation 47
3.3.1 Network Model 47

3.3.2 Association and Transmission Model 48
3.3.3 Queuing Model 50
3.3.4 Scheduling and Gain Model 50
3.3.5 Problem Formulation 51
3.3.6 Reduced-complexity Scheduling Model 53
3.4 DRL-based Throughput and Delay Optimality 54
3.4.1 DRL for Multi-objective Optimization 54
3.4.2 Preliminaries of POMDP 57
3.4.3 Optimize with Lyapunov Function 61
3.5 Algorithm Design 63
3.5.1 DDQN-based Solution 65
3.5.2 Implementation of DDQN-based Algorithm 66
3.6 SIMULATION RESULTS 70
3.6.1 Simulation Setup 70
3.6.2 Result Analysis 71
3.7 Conclusions 78

4 QoS-guaranteed Routing Protocol for Intelligent Transportation System: A Hybrid Approach 79

4.1 Introduction 80
4.2 Related Work 83
4.2.1 Distributed Solutions 83
4.2.2 Centralized Solutions 83
4.3 System Model 84
4.3.1 Network Architecture 84
4.3.2 Assumptions and Network Settings 86
4.3.3 Network Model 87
4.3.4 Channel Model 88
4.3.5 Delay Model 89

4.3.6 Connectivity Model 90
4.4 Problem Formulation 91
4.4.1 Network Clustering 91
4.4.2 Route Planning 93
4.4.3 In-network Control 93
4.5 QoS-guaranteed Routing Protocol 95
4.5.1 Predictive Clustering Algorithm with Feedback Control 95
4.5.2 QoS-guaranteed routing algorithm 97
4.5.3 Residual link lifetime-based relay selection 99
4.5.4 Complexity and Control Overhead Analysis 102
4.6 Performance Evaluation 103
4.6.1 Simulation setting 104
4.6.2 Clustering algorithm performance verification 105
4.6.3 Global route planning performance verification 106
4.6.4 QCRP performance verification 108
4.7 Conclusion 111

5 QMAC: Resource Allocation for Advanced Vehicular-to-Everything (V2X) Application 112

5.1 Introduction 112
5.2 Related Work 115
5.2.1 5G NR-V2X 115
5.2.2 Resource Allocation Solutions in V2X 116
5.3 System Model 117
5.3.1 Scenario Description 117
5.3.2 Network Model 119
5.3.3 Channel Model 119
5.3.4 Reliability Model 121
5.4 Problem Formulation 122

5.4.1 Resource Efficiency Optimization 122
5.4.2 Power and Error Control Selection Game 124
5.5 QoS-guaranteed Medium Access Control 125
5.5.1 Branch and Bound-based Centralized Resource Allocation at BS 126
5.5.2 Control Strategy Adjustment with Bandit Feedback 128
5.5.3 Complexity Analysis 132
5.6 Performance Evaluation 133
5.6.1 Experiment setting 134
5.6.2 Data rate performance analysis 135
5.6.3 Reliability performance analysis 136
5.6.4 Resource efficiency performance analysis 138
5.7 Conclusion 140

6 Summary **141**
6.1 Appendix for Chapter 2 144
6.2 Appendix for Chapter 3 145
6.3 Appendix for Chapter 4 147

Bibliography **149**

List of Tables

Table 2.1 Notations and definitions 11
Table 2.2 Network configurations 33
Table 2.3 Experiment settings 34

Table 3.1 Notations and definitions 44

Table 4.1 Notations and definitions 80
Table 4.2 Mean square error (MSE) of four prediction models: Gaussian Process Regression (GPR), Linear Regression (LR), Support Vector Regression (SVR), and LSTM predicting vehicles' locations in 10 time slots with different speed. 95
Table 4.3 Parameters Setting 104
Table 4.4 Connectivity performance comparison 106

Table 5.1 Notations and definitions 113
Table 5.2 Parameters Setting 134
Table 5.3 Reliability Performance Comparison 137

List of Figures

Figure 1.1 SET protocol architecture [21] 4
Figure 1.2 Structure of the thesis. 5

Figure 2.1 Router structure overview 16
Figure 2.2 System model overview 18
Figure 2.3 DSRP-enabled network with finite buffers 19
Figure 2.4 Numerical analysis of DSRP throughput improvements. 20
Figure 2.5 Network topologies used in the experiments. 32
(a) 3×3 grid topology. 32
(b) U.S. backbone network. 32
Figure 2.6 Comparisons under bursty traffic 35
(a) CDF of per-packet E2E delay. 35
(b) Average QoS performances. 35
Figure 2.7 QoS performances with high link rate 36
(a) Goodput performance. 36
(b) E2E delay performance. 36
Figure 2.8 QoS performance comparison using mesh network 37
(a) Average delay performance. 37
(b) Goodput performance. 37
Figure 2.9 Comparisons under different delay budgets. 37
(a) Average E2E delay. 37
(b) Goodput. 37

Figure 2.10QoS performance comparisons on DiffServ property for DS applications. 39
(a) DS application: Average delay. 39
(b) DS application: Goodput. 39
Figure 2.11QoS performance comparisons on DiffServ property for NDS applications. 40
(a) NDS application: Average delay 40
(b) NDS application: Goodput. 40
Figure 2.12Performance gain achieved by different algorithms. 41

Figure 3.1 Delay-aware selective scheduling. 48
Figure 3.2 Implementation of priority-based single-queue packet selection . 53
Figure 3.3 Illustration of the decomposition and parameter-transfer strategy. 56
Figure 3.4 Illustration of DDQN framework for resource scheduling in networks. 61
Figure 3.5 Training process of DDQN-based algorithm for solving P1 and P2. 67
Figure 3.6 Pareto-Front achieved by the proposed DRL-based framework. . 71
Figure 3.7 Convergence of different algorithms for the overload case. . . . 72
Figure 3.8 Two cases of delay versus traffic intensity. 73
(a) Inner capacity case. 73
(b) Overload case. 73
Figure 3.9 Delay versus traffic intensity for different design schemes. 74
Figure 3.10Drop rate versus traffic intensity for the overload case. 75
Figure 3.11Drop rate versus traffic intensity for different design schemes. . 76
Figure 3.12Goodput performance under different traffic intensity. 77
(a) Goodput of different schemes. 77
(b) Goodput of each queue. 77

Figure 4.1 An overview of network architecture. 85
(a) QCRP protocol architecture 85
(b) Network architecture for ESS applications in vehicular networks 85
Figure 4.2 In-network control. The original path fails the transmission distance constraints after t time slots due to mobility. LPCA performs re-routing to locally update the routing path. 94
Figure 4.3 Trellis graph representation 97
Figure 4.4 Residual link lifetime evaluation 100
Figure 4.5 Clustering performance comparison under high traffic density scenario. 105
(a) Short-term performance 105
(b) Long term performance 105
Figure 4.6 Global routing algorithm comparison without local control. . . 107
(a) Average E2E delay comparison 107
(b) Average E2E PDR comparison 107
Figure 4.7 QCRP E2E performance comparison: first row under high traffic density and second row under sparse traffic density 109
(a) Average E2E throughput comparison (High) 109
(b) Average E2E throughput comparison (Sparse) 109
(c) Average E2E PDR comparison (High) 109
(d) Average E2E PDR comparison (Sparse) 109
Figure 4.7 QCRP E2E performance comparison: first row under high traffic density and second row under sparse traffic density 110
(e) Average E2E latency comparison (High) 110
(f) Average E2E delay comparison (Sparse) 110

Figure 5.1 An overview of the extended sensor sharing application on a multi-lane highway scenario, where dedicated bandwidth is divided for V2I and V2V communications. 118

Figure 5.2 An example for periodic resource reservation considering HARQ retransmission. 122
Figure 5.3 Overview of the proposed hybrid control strategy: centralized control at BS and distributed control at relay vehicles. 126
Figure 5.4 MAB-based control strategy adjustment. The relay vehicle evaluates the previous control strategy during the sensing window and configures the new strategy in the selection window. 129
Figure 5.5 Data rate performance comparison under different traffic densities. 135
(a) Data rate performance under high traffic density 135
(b) Data rate performance under low traffic density 135
Figure 5.6 Reliability performance comparison under different traffic densities. 137
(a) High traffic density 137
(b) Low traffic density 137
Figure 5.7 Resource occupancy ratio comparison under different traffic densities. 138
(a) High 138
(b) Medium 138
(c) Low 138
(d) Sparse 138
Figure 5.8 Network network utility performance comparison under different traffic densities. 139
(a) High traffic density 139
(b) Low traffic density 139

ABSTRACT

The success of recent communication and computing technologies has attracted growing interest from both academia and industry in developing advanced applications such as metaverse, digital twin, and intelligent transport systems, which are characterized by ultra-high reliability and low latency communication (URLLC) requirements. More importantly, they require guaranteed services and their performance will degrade quickly if any metric is not satisfied, However, the existing routing and scheduling solutions may not fully support these applications. We observe one major reason is the lack of collaboration between each layer in the protocol stack and inefficient handling of network information.

To bridge the gap, we leverage a novel protocol architecture SET which enables flexible protocol assembling via control function decomposition. A new concept called protocol control agent (PCA) is introduced to enable in-network intelligence and enhanced adaptability. PCA leverages cross-layer network information and collaborations to support QoS requirements and improve resource efficiency. In addition, we consider the freshness of network information for designing new protocols based on SET. To this end, we develop a range of scheduling and routing solutions to support ultra-reliable and low-latency communications in different networks.

We start with the end-to-end delay guarantee problem in wired data networks. A novel distributed solution named delay-guaranteed scheduling and routing protocol (DSRP) is proposed to provide packet-level delay guarantees and differentiated services within an autonomous system. In DSRP, we adopt priority queues with fixed buffer sizes to guarantee per-hop delay bounds and serve traffic with different delay requirements, and leverage path diversity for multiplexing gains. To address the congestion issue, we introduce a congestion-aware mechanism where neighboring nodes exchange queue information to reflect their congestion status. Given the rich routing decision space, a scheduling algorithm based on renewal optimization named DSROpt is proposed to maximize overall network utility.

Next, we investigate the scheduling problem for time-critical applications in a

single-hop downlink wireless network. For time-critical applications, being late is as severe as being dropped. However, packets are normally left with a small delay budget at the last hop and are likely to exceed their delay requirement when there is a standing queue. To address this issue, we propose a delay-aware selective scheduling (DASS) algorithm that selects packets to serve. In DASS, we introduce a new output gain model based on delay laxity and packet priority for characterizing each scheduling decision and formulated a multi-objective optimization problem to determine the optimal scheduling policy. Due to the problem's complexity and uncertainty of the network environment, a deep reinforcement learning framework is proposed to find the Pareto optimal scheduling decisions that maximize network utility and guarantee per-packet delay requirement.

While many new applications operate in mobile networks, it is crucial to design QoS-guaranteed solutions with the consideration of more challenges such as mobility and dynamic channel conditions. Thus, we extend our research to the vehicular network. To address these issues, we propose a hybrid control framework leveraging global and local network information and taking advantage of the time-scale difference between the change rate of network typology and channel condition to perform control at different scopes.

Based on the framework, we first focus on the routing problem in vehicular networks. A QoS-guaranteed clustering and routing protocol (QCRP) is proposed. In QCRP, the global information, i.e., network topology is used for clustering and route planning to ensure per-cluster connectivity and routing path initialization while the local information, i.e., channel condition and vehicle location are used for re-routing. Next, to support efficient in-network communication under the constraints of limited network resources, a QoS-guaranteed medium access control (QMAC) protocol to perform resource allocation is required. Following the same framework, a centralized spectrum allocation combined with distributed power and error control solution is proposed.

Overall, in this thesis, we introduce a new perspective of supporting stringent

QoS requirements and demonstrate the effectiveness of our solution in various network settings. Our work opens up new possibilities for research extensions and applications.

ACKNOWLEDGEMENTS

I want to thank:

my supervisor and mentor, Dr. Lin Cai, her guidance, support, and patience make my Ph.D. journey invaluable. Her wisdom, rigor, and passion for science have consistently inspired me in my research, career, and life. I feel incredibly fortunate to be her student. Special thanks to Dr. Jianping Pan, although he did not directly supervise me, he has influenced me with his words and actions. They are my role models both in my career and life.

my supervisory committee and external examiner, Dr. Kin Fun Li, Dr. Sudhakar Ganti, and Dr. Qiang Ye for spending their precious time reviewing my thesis and attending my oral exam.

my labmates in both CNLAB and PANDA for all the wonderful times we shared. Special thanks to Dr. Yue Li, who helped me understand the essence of good leadership; Dr. Jiequ Ji, who showed me how to become a productive researcher; and Dr. Seyed Hamed Mosavat-Jahromi, who inspired me to persevere through life's hardships.

my friends and colleagues for all the sweet and bitter memories we went through together. They are the shining stars accompanying my journey to become a better person.

my family, for their unselfish love and care. Specially my parents, have always been the greatest support throughout my life. I am, and always will be, grateful and proud to be your son.

I would like to end with one of my favorite quotes from a movie, *The Wind Rises*, by Hayao Miyazaki:

Le vent se lève, il faut tenter de vivre

纵有疾风起，人生不言弃

DEDICATION

To my mom, dad, and myself.

Chapter 1

Introduction

1.1 Background

The development of communication and computing technologies is spurring new applications with advanced features. The fifth generation (5G) communication technology attracted great attention and quickly, has been deployed worldwide since its standardization by the 3rd Generation Partnership Project (3GPP). It is anticipated that the 5G application market alone will reach hundreds of billion U.S. dollars by 2030 [7]. In the meantime, the next-generation communication technology (6G) has been actively researched [57]. Several projects have been launched led by 3GPP and top technology companies [2]. New applications such as metaverse, digital twin, extended reality, and intelligent transport systems require stringent and guaranteed quality-of-service (QoS) over latency, data rate, and reliability. These services are characterized as ultra-reliable and low latency communication (URLLC). For the applications requiring URLLC services, a single aspect of QoS failure can result in significant performance degradation.

However, URLLC is challenging to achieve using conventional scheduling and routing solutions [64,120,127]. Traditional data networks are designed for applications requiring best-effort services and often experience packet loss and long latency caused by network congestions or software/hardware failures [4,23]. The existing network archi-

tecture adopts the layered and end-to-end design principles to achieve good modality, scalability, and robustness. But some of these features are not insufficient to support URLLC service. For example, redundant control in each layer adds additional latency and the randomness of network dynamics is very difficult if not impossible to handle using single-layer control. Therefore, scheduling and routing with cross-layer collaborations are critical in achieving URLLC.

In addition, the distinctions among packets with different priorities, e.g., delay requirements, are often ignored in the traditional network where all packets are treated equally. Such design is unsuitable for the co-existence of time-critical applications and delay-tolerant applications. It is crucial to introduce priorities to networks and provide differentiated services (DiffServ) to different applications to achieve high network utilization. In addition, for time-critical applications, packets arriving late is as severe as being dropped. Thus, a delay-guaranteed service at the packet level is desired.

In recent years, the deterministic network (DetNet) led by the IETF DetNet working group has become a hot research topic. DetNet aims at providing delay-guaranteed services on a per-deterministic-flow basis by exploring explicit data paths [74, 90] using technologies such as multiprotocol label switching (MPLS), software-defined network (SDN), network slicing, and IEEE 802.1 Time-Sensitive Networking (TSN) [19, 49, 133] in a centralized manner. However, such resource reservation-based solutions guarantee the end-to-end delay for each traffic flow at the cost of losing multiplexing gain and suffering from efficiency and scalability issues.

In addition to static and wired networks, both academia and industry are interested in guaranteeing QoS in wireless networks. The emerging vehicular network, for example, has attracted great attention as it is anticipated to be the key technology to enable autonomous driving [11, 120]. However, guaranteeing QoS in vehicular networks is more challenging given the dynamic characteristics of wireless channels and user mobility. In addition, spectrum scarcity, signal interference, and limited computing capacity bring more difficulties in achieving QoS-guaranteed communication.

The scheduling and routing problems in vehicular networks have been heavily discussed in the literature for general vehicular-to-everything (V2X) applications [13, 122]. A software-defined vehicular network (SDVN) has been introduced, which integrates the concept of software-defined network (SDN) into vehicular networks for higher flexibility and programmability [62, 134]. In the recent 5G New Radio (NR) V2X release, a range of methods have been defined to enhance V2X communications [43]. However, the existing works are insufficient or inefficient in guaranteeing the required QoS specified in 3GPP advanced V2X applications [55]. In this regard, how to guarantee these QoS requirements under the constraints of spectrum, power, and computing limitations remains an open issue.

1.2 Design Principle of New Protocols for Ultra-Reliable-and-Low-Latency Communications

The above challenges motivated us to study how to efficiently guarantee the QoS requirements in future networks under different scenarios. We observe that it is very difficult, if not impossible, to guarantee QoS in a distributed fashion using single-layer information or local information alone. On the other hand, a centralized solution can be inefficient and lacks scalability to guarantee QoS using the global network information. Furthermore, developing specific protocols for each new application is too costly given the diverse QoS requirements and heterogeneity of future communication systems.

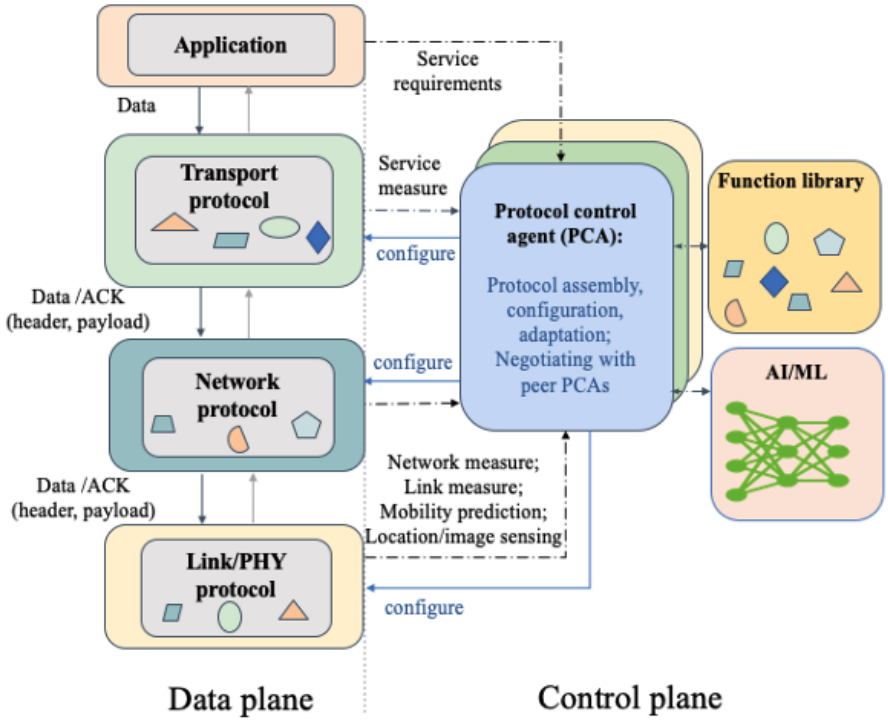


Figure 1.1: SET protocol architecture [21]

To address these challenges, [21] proposed a Self-Evolving and Transformative (SET) protocol architecture that offers high flexibility via control function decomposition and reassembling. By leveraging the modular design principle, SET architecture reassembles the protocol based on needs to support different types of services. As shown in Fig. 1.1, a novel concept named protocol control agent (PCA) resides in the control plane is introduced in SET to break the boundaries in the layered protocol architecture. By leveraging the decomposed network control functions (contained in the function library in Fig. 1.1) and advanced learning algorithms, PCA selects and configures protocols based on the observed network information and application requirements carried in each packet. For example, for each packet/flow, PCA starts with a default control protocol according to the application's service requirement. During the transmission, PCA observes the achieved QoS and network condition. If the QoS requirements are not satisfied, e.g., PCA observes an increase in packet loss, a new control protocol at a higher cost, e.g., adding an error control to the packet/flow,

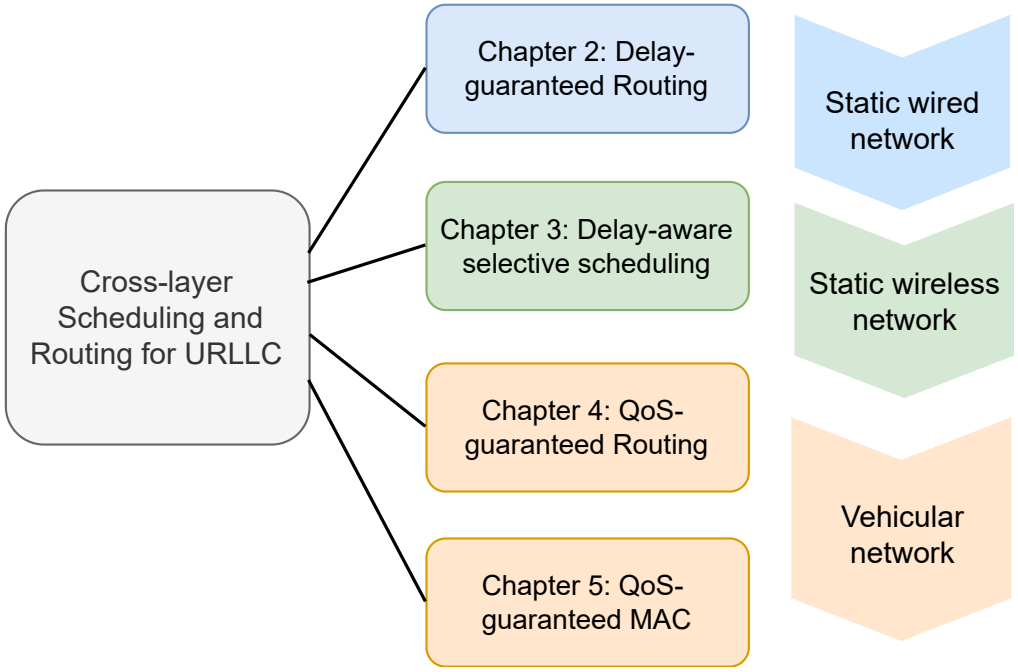


Figure 1.2: Structure of the thesis.

will be adopted to maintain the required QoS. More details of SET architecture and PCA are introduced in [21].

In addition, we observe that network information shows different *freshness*. For example, some network information such as QoS requirements, network settings, and network performance bounds, etc. are less variant, i.e., have better freshness. In other words, they do not need frequent updates to ensure network performance. Thus, they are suitable to be exchanged globally and utilized for centralized control. Other control information with poor freshness, such as link condition and queue status change rapidly over time and may become invalid after traversing through the network. Thus, they are more suitable for distributed local control.

Following this design principle and SET architecture, in this thesis, we explore the possibilities of new scheduling and routing solutions using cross-layer information and leveraging in-network collaborations to guarantee QoS performances in various networks. The thesis structure is shown in Fig. 1.2.

To begin with, we first research the scheduling and routing problems in static

networks without considering network mobility. In Chapter 2, we investigate a novel delay-guaranteed solution at the packet level in wired data networks where information from multiple layers at different scales is utilized for making scheduling and routing decisions. Since the link condition almost remains constant over time in wired networks, in Chapter 3, we take one step further by studying an efficient downlink scheduling solution in wireless networks. The scheduler in the base station selectively transmits packets based on priorities and channel conditions.

Next, we consider a more challenging scenario in vehicular networks where both network mobility and wireless channels are considered. In Chapter 4, we propose a predictive clustering algorithm based on global network topology to ensure long-term network connectivity and within each cluster, an adaptive routing algorithm is proposed to guarantee QoS performance leveraging local information. However, given the scarcity of spectrum resources and unstable channel quality, it is impossible to achieve QoS-guaranteed routing in vehicular networks without efficient resource allocation. Therefore, we investigate a flexible medium access control protocol via centralized spectrum allocation and distributed power and error control in Chapter 5.

1.3 Proposed Research Issues and Contributions

1.3.1 Congestion-aware delay-guaranteed scheduling and routing with renewal optimization

In Chapter 2, we investigate a scheduling and routing solution to guarantee the end-to-end delay in wired data networks. A novel distributed, delay-guaranteed, and congestion-aware network architecture called delay-guaranteed scheduling and routing protocol (DSRP) that provides DiffServ to various applications and guarantees end-to-end delay is proposed. In DSRP, each time-critical packet carries a maximum tolerable delay requirement called *delay budget*, and the network is responsible for delivering the packet within its delay budget using information collected from both the network

layer and link layer. Moreover, to reduce network congestion due to bursty traffic, neighbouring routers exchange their queue information periodically so the upstream node can adjust forwarding decisions in advance. The main contributions of this chapter can be summarized as follows:

- We address the delay-guaranteed service problem using a cross-layer approach, where the link layer queue management and the network layer routing work together to ensure packet-level end-to-end delay.
- We propose a novel scheduling and routing algorithm based on renewal optimization. The proposed algorithm is run by each network node which selects a suitable forwarding decision for each packet using delay requirement, queue status of the current node, and the congestion status of the downstream nodes.
- We build a prototype using the ns-3 simulator. Extensive simulations are conducted to evaluate the proposed solution compared with the state-of-the-art and baseline approaches. DSRP achieved significant goodput improvements and high network utility while guaranteeing the E2E delay.

1.3.2 Downlink Scheduler for Delay Guaranteed Services Using Deep Reinforcement Learning

In Chapter 3, we study the downlink resource scheduling problem in a general wireless network. A novel delay-aware selective scheduling policy is proposed to guarantee the delay requirement of time-critical packets and achieve network utility optimality. To determine the optimal policy, a multi-objective optimization problem is formulated aiming to minimize the average queue length while maximizing the network utility of the system under the constraints of guaranteeing per-packet delay and achieving fairness among users. The main contributions can be summarized as follows:

- We propose a delay-aware scheduling policy for random arriving packets with different delay requirements, where early delay-outage dropping is considered.

Moreover, we define an output gain function that combines the delay laxity and priority of different packets to show the penalty for dropping packets and the reward for successful transmission.

- We formulate a multi-objective optimization problem that minimizes the average queue backlog while maximizing the average output gain under the constraints of achieving fairness among users and guaranteeing per-packet delay. A deep reinforcement learning framework is proposed to solve this intractable problem via decomposing it into a set of subproblems.

- Simulation results show that our proposed solution converges to a Pareto optimal policy faster than the comparison solutions and achieves significant reductions in average delay and delay outage drop rate.

1.3.3 QoS-guaranteed Routing Protocol for Intelligent Transportation System: A Hybrid Approach

In Chapter 4, we focus on the routing problem in vehicular networks. We present a QoS-guaranteed clustering and routing protocol (QCRP). Two types of protocol control agent (PCA), global PCA (GPCA) and local PCA (LPCA) are defined in QCRP following the SET architecture. GPCA uses the global network information to cluster the network and find the suitable routing path to satisfy the E2E QoS requirements. LPCA uses the local network information to fine-tune control decisions to adapt to network dynamics. Moreover, topology control and re-routing are adopted to reduce control overhead and maintain network performance over time. In summary, the contributions of this chapter are threefold:

- We introduce a novel flexible protocol architecture to provide guaranteed QoS for various applications while adapting to network dynamics.

- We propose a novel protocol named QCRP to support advanced V2X application which requires stringent E2E QoS for multiple receivers in a high mobility

environment.

- We show that both global control and local control are essential to maintaining the E2E QoS performance over time via extensive simulation experiments, where the former enables control optimality while the latter contributes to network dynamics adaptation.

1.3.4 QMAC: Resource Allocation for Advanced Vehicular-to-Everything (V2X) Application

In Chapter 5, we focus on the resource allocation problem in vehicular networks. We propose a QoS-guaranteed medium access control (QMAC) protocol combining both centralized and distributed control decisions leveraging the off-the-shelf tools defined in 5G NR-V2X standards and a lightweight online learning algorithm. QMAC is designed to optimize the overall resource efficiency while guaranteeing the QoS requirements. The contributions of this chapter can be summarized as follows:

- We investigate the resource allocation problem of supporting stringent QoS requirements with limited spectrum resources in vehicular networks. A hybrid control solution named QMAC that flexibly utilizes the existing 5G NR-V2X methods is designed to improve resource efficiency.

- We develop a branch-and-bound-based algorithm for centralized spectrum allocation. As for distributed power control and error control selection, we consider a non-communicative scenario where vehicles make control decisions independently using an online learning algorithm based on channel-state information (CSI) feedback from the receivers.

- We verify our solution via extensive simulation experiments. The results show that QMAC can achieve the highest resource efficiency while guaranteeing the QoS requirements under different traffic densities.

Chapter 2

Congestion-aware Delay-guaranteed Scheduling and Routing with Renewal Optimization

The next-generation networks are facing new challenges and requirements characterized by stringent QoS requirements such as extremely low latency and extremely high reliability [88, 127]. Typical applications including metaverse, digital twin, and real-time control involve large volumes of both delay-sensitive (DS) and non-delay sensitive (NDS) data exchange [59, 64, 115]. Although bandwidth over-provisioning has been adopted as an engineering solution to satisfy the stringent QoS requirements, how to effectively and efficiently support both DS and NDS applications remains an open issue [35, 116]. In this chapter, we focus on the routing and scheduling problem in wired data networks to provide end-to-end (E2E) delay-guaranteed service for DS applications while ensuring compatibility with NDS applications.

Table 2.1 summarizes the notations and definitions frequently used in this chapter.

Table 2.1: Notations and definitions

Symbol	Definitions
$\mathcal{N}$	Set of nodes
$\mathcal{K}$	Set of priority queues
$\mathcal{H}$	Set of forward decision
$\mathcal{P}$	Set of profits achieved by each queue
T_{budget}	End-to-end delay requirement (delay budget)
R	Link rate
T_{WRR}	Total service time of each round-robin period
T_k	Delay upper bound of queue k
$\hat{\mathcal{H}}$	Set of simplified forward decision
$h_{i,j}^k$	Forward decision for each packet, $i, j \in \mathcal{N},\ k \in \mathcal{K}$
$Q_{i,j}^k[t]$	Queue length of queue k at port j of node i at time slot t
B_k	Buffer size of queue k
w_k	Portion of service time assigned to queue k
GI	Delay upper bound of the highest priority queue
$\text{Cost}(\cdot)$	Routing cost (delay) of a path
$g_d(\cdot)$	Network utility function for delay-sensitive packet
$g_n(\cdot)$	Network utility function for non-delay sensitive packet
T_{budget}^*	Residual delay budget
T_{hop}	Per-hop delay budget
α	Control parameter for queue management
γ	Congestion index
$G_{i,j}^k[t]$	Output gain of $h_{i,j}^k$ at time slot t
$T_{i,j}^k[t]$	Time cost of $h_{i,j}^k$ at time slot t
θ	Reward per unit time cost ratio
$\mathcal{D}(\cdot)$	Complete cost-reward pair set
L	Delay laxity
P_k	Profit achieved using queue k

2.1 Introduction

The routing and scheduling problem to provide end-to-end (E2E) delay-guaranteed service for DS applications has been extensively studied from different aspects. In recent years, the deterministic network (DetNet) led by the IETF DetNet working group has become a hot research topic to provide delay-guaranteed services on a per-deterministic-flow basis by exploring explicit data paths [75,91]. The DetNet takes the

joint effort of both layer three routed segments and layer two bridged segments using technologies such as multiprotocol label switching (MPLS), software-defined network (SDN), and IEEE 802.1 Time-Sensitive Networking (TSN) [19,50,133] in a centralized manner. The data traffic of each deterministic flow is delivered with guaranteed delay and low delay variation constraint via resource reservation [44]. To find the optimal routes with guaranteed delay in a given network, several routing algorithms have been proposed using the worst-case delay [5, 16, 63]. However, the existing approaches have various limitations. Resource reservation-based solutions guarantee delay at the cost of multiplexing gain. The worst-case delay-based routing algorithms can be too conservative to serve DS applications with low latency requirements. The flow-based traffic engineering solutions face scalability issues and cannot guarantee delay at the packet level. In addition, these solutions are unaware of the deadlines of each packet when making routing decisions and fail to handle congestion caused by network dynamics.

On the other hand, the distributed E2E delay-guaranteed solution at the packet level is underexplored given the following challenges. First, it is difficult to handle large variations in the E2E delay. The highly dynamic feature in E2E delay is mainly caused by the varying queuing delay at each hop in the network and is difficult to predict due to the randomness in network traffic. Second, the diversity in delay requirements. For most DS applications, packets belonging to the same flow can have different priorities or delay requirements. For example, I/B/P frames have different importance for MPEG videos [78]. How to effectively guarantee various delay requirements at the same time, i.e., satisfying high-priority packets without starving the low-priority packets, remains a critical task. Last but not least, how to efficiently handle the bursty traffic while guaranteeing delay with good load balance is challenging. The traditional shortest-path routing algorithms can result in heavy congestion in certain paths due to bursty traffic causing long queuing delays and high packet loss rates.

To bridge the gap, we propose a novel distributed, delay-guaranteed, and congestion-

aware network architecture called delay-guaranteed scheduling and routing protocol (DSRP) which provides DiffServ to various applications and guarantees the packet-level end-to-end delay. Specifically, each DS packet carries a maximum tolerable delay requirement called *delay budget* specified by the source application. The network is responsible for delivering the packet within its delay budget using information collected from both the network layer and the link layer. The priority queue is adopted to provide DiffServ to packets with different delay requirements. Given the randomness of network traffic, we rely on the per-hop delay upper bound to guarantee the E2E delay, where each priority queue is assigned a fixed buffer size and service rate. Moreover, to avoid congestion due to bursty traffic in the network, neighboring routers exchange their congestion information periodically so the upstream node can adjust routing decisions in advance.

In addition, we develop a scheduling and routing solution named *DSROpt* that makes routing decisions and schedules packets to different priority queues by jointly considering their delay budgets, queuing delay, and neighborhood congestion status. To this end, we formulate an optimization problem that maximizes the overall network utility under the constraint of the E2E delay requirement of each packet. We solve the problem in a fully distributed manner where each node determines a suitable forward decision for each packet based on the renewal optimization theory.

The rest of the chapter is organized as follows: Section 2.2 introduces the related work. Section 2.3 presents the system model and problem formulation. The design details and working mechanisms of DSRP are explained in Section 2.4. In Section 2.5, the simulation settings and experimental results are explained and analyzed. Finally, conclusions and future research issues are discussed in Section 2.6.

2.2 Related works

2.2.1 Delay-guaranteed Network Architecture

In [33], a latency-based forwarding (LBF) solution was proposed to achieve high-precision latency objectives. LBF focused on a link layer forwarding strategy where each node takes different actions on receiving the packet. LBF consists of two major parts, i.e., latency budget determination and QoS Action. The actions taken at each node to guarantee delay depend on the packet's remaining latency budget, destination, and the latency encountered. The Push-In First Out (PIFO) queue was adopted to insert the packet to a specific position that satisfies the latency budget's lower bound. However, the LBF did not involve routing and requires prior knowledge of the routing path for each packet, which may cause delay guarantee failure due to network dynamics or router failure. In addition, the PIFO enqueue strategy applied in LBF may cause strong decision interference among successive packets and result in longer delays than expected.

An adaptive routing protocol was proposed in [3]. The solution assumes that the remaining delay budget for each arriving flow is known to the network. The delay upper bound and the typical delay of each link are exchanged in the network to generate a lookup table indicating the guaranteed delay bound to reach the destinations. In the run-time phase, the node selects the path with the smallest typical delay and checks whether the delay budget of the path is within the remaining delay budget. However, the typical delay is highly dynamic. It either requires a high overhead to collect this information, leading to high oscillations of path selection and congestion; or the typical delay used in the algorithm is inaccurate leading to poor path selection.

2.2.2 Delay-based Scheduling and Routing Protocol

The backpressure (BP) routing algorithm has been widely adopted to achieve throughput optimality by stabilizing the network [71, 100]. The original BP algorithm used

the backlog gradient to formulate the throughput problem, which, however, suffers from severe queuing delay. To address this problem, many extensions of the BP algorithm have been developed in terms of different congestion gradient metrics. In [52], a sojourn time-based BP algorithm (STBP) was proposed. The accumulated sojourn time of buffering packets describes the congestion status of each queue instead of queue length. The sojourn time backlog measurement has the advantages of faster increment, reducing random walk packet delay and last packet delay.

More recently, a routing algorithm that takes the advantages of both BP and max-weight scheduling algorithms named MW+BP was proposed to achieve overload balancing [132] in a single-hop network with bounded buffers, where the max-weight part aims to serve longer queues while the BP part balances the load between ingress and egress buffers. However, the existing approaches are insufficient to achieve delay-guaranteed service at the packet level. First, the diverse delay requirements of different packets of the same flow are not considered. Second, the temporal correlation between scheduling decisions is not fully explored. Last, considering the queue delay of the current node only is insufficient because the packet may be dropped due to congestion in the next hop.

2.3 System Model and Problem Formulation

2.3.1 Network Architecture

Router Structure

The per-hop delay in a network is mainly composed of queuing delay, processing delay, transmission delay, and propagation delay. We focus on the queuing delay as it is the major cause of the E2E delay variation of packet delivery. While the instantaneous queuing delay varies quickly due to high uncertainties in packet arrival

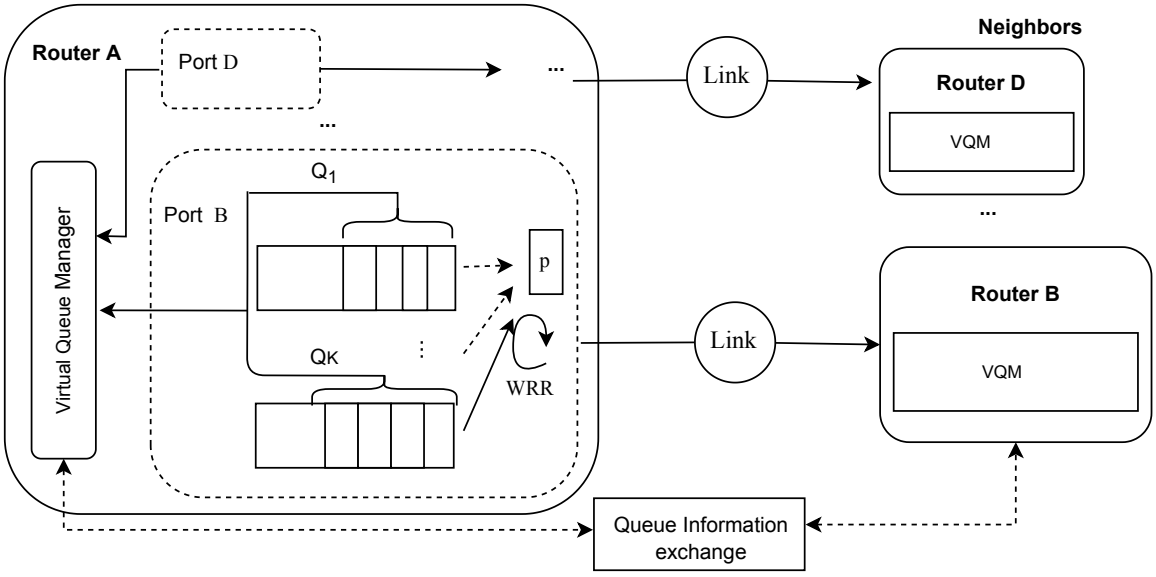


Figure 2.1: Router structure overview

rate, we adopt the queue delay upper bound for route calculations. Each router in the network maintains the delay upper bound of each queue by fixing the buffer size and service rate. In addition, it exchanges its local queue information with neighbors to advertise congestion information. An overview of the router structure is shown in Figure 2.1.

Each port of the router is deployed with a set of priority queues to provide Diff-Serv for DS and NDS packets. The queues can be classified into two categories in terms of service types, namely delay-guaranteed (DG) queues, and best-effort (BE) queues. DG queues provide delay-guaranteed services for the DS packets while BE queues only serve NDS packets without any delay guarantee. Using multiple priority queues for DS packets helps to provide finer granularity of delay-guaranteed services and a rich forward decision set. To avoid always starving low-priority queues, the weighted-round-robin (WRR) scheduling approach [143] is adopted where each queue is assigned with a certain portion of service time, i.e., weight, for packet transmission. Thus, the per-hop delay of each DG queue is upper bounded by fixing the buffer size and weight of each queue[1]. Note that in DSR, we assume a shallow buffer size for

[1]For example, for a router with the link rate of R Mbps requiring a queue delay upper bound of

each DG queue to guarantee a tight delay upper bound[2]. Since the BE queue does not guarantee a delay upper bound, its buffer size can be sufficiently large to store more NDS packets.

However, bounding the per-hop delay alone is insufficient to guarantee the E2E delay performance because packets may be lost due to network congestion [51]. In this chapter, we address network congestion in the network layer via neighbor routers information exchange and re-routing. A virtual queue manager (VQM) is designed for collecting and exchanging queue information with neighbors as shown in Figure 2.1. In general, VQM in each router collects the local queue information, i.e., the queue length of each priority queue, and exchanges it with neighbors. The queue information essentially reflects the congestion status of each router. In this case, routers are aware of network congestion and are able to adjust forward decisions for each packet in advance before congestion becomes severe.

System Model

The system model of the proposed DSRP network architecture is shown in Figure 2.2. We assume every DS packet carries a delay budget in its packet header before being injected into the network. The delay budget is used by routers to make routing and scheduling decisions to guarantee the E2E delay requirements. In the meantime, routers monitor their local queue information while exchanging two types of information with each other, i.e.,

- Link-state information including delay upper bound of the highest priority queue (denoted by GI)[3]. GI is broadcast in the entire network similar to the link state information used in the open-shortest-path-first (OSPF) routing pro-

T_k ms in queue k, the corresponding buffer size should be at most $B_k = 0.125 w_k R T_k$ KB, where w_k is the weight allocated to queue k.

[2]We assume a light weight of DS traffics in the network and the tradeoff between buffer size and packet drop is beyond the scope of this chapter.

[3]The highest priority queue is chosen because it is the minimum per-hop delay that can be guaranteed by each router and it allows a finer granularity for path selection, especially for urgent packets with small delay budgets.

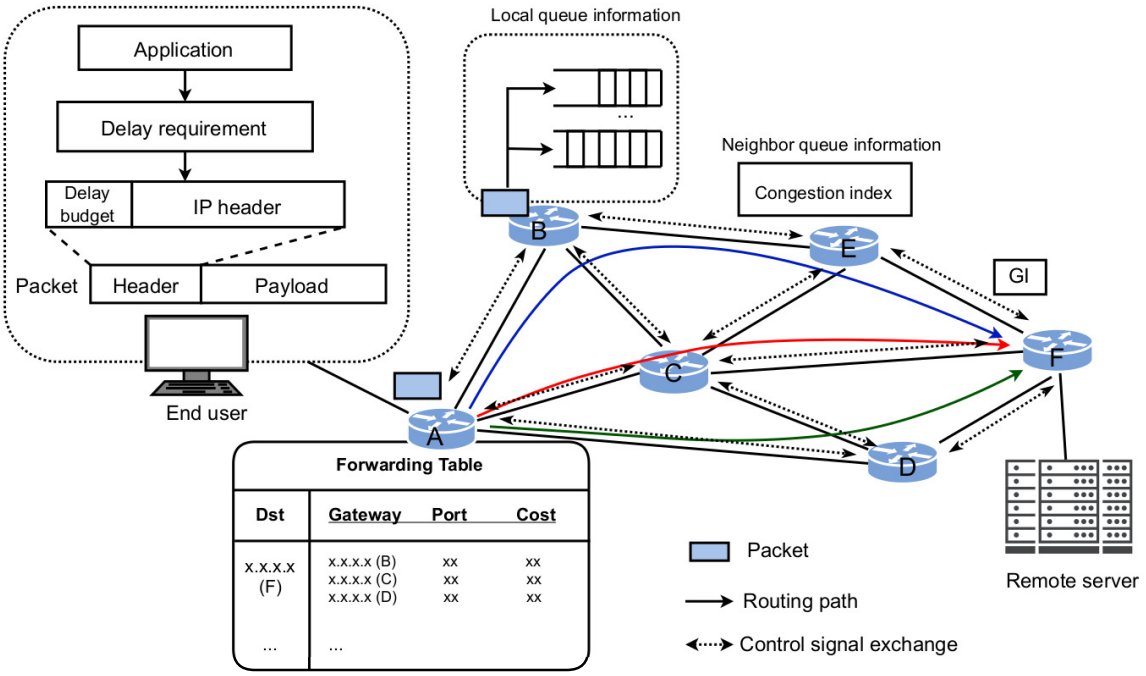


Figure 2.2: System model overview

tocol [37].

- Neighborhood queue information (NI), i.e., the local queue information shared by neighboring routers. NI is useful for route selection and congestion avoidance. Note that NI is exchanged more frequently at the level of a few milliseconds[4].

Routers in the DSRP network generate forwarding tables based on GI. It is worth noting that unlike single-path link state routing protocols such as OSPF, DSRP-enabled routers adopt multipath exploration to reach the destination by computing the shortest paths rooted from their neighbors. For example, for a target pair from A to F as shown in Figure 2.2, three shortest paths can be obtained at router A by running the shortest path algorithm from router B, C, and D, respectively, based on the link-state information.

In our routing algorithm, multipath exploration provides larger routing decision space at the cost of increment in time and space complexity. In a network consists of

[4]For easier implementation in practice, NI of each router can be a few bits piggybacked in the frame header. In this context, the additional bandwidth cost of NI exchange is only a few Kbps.

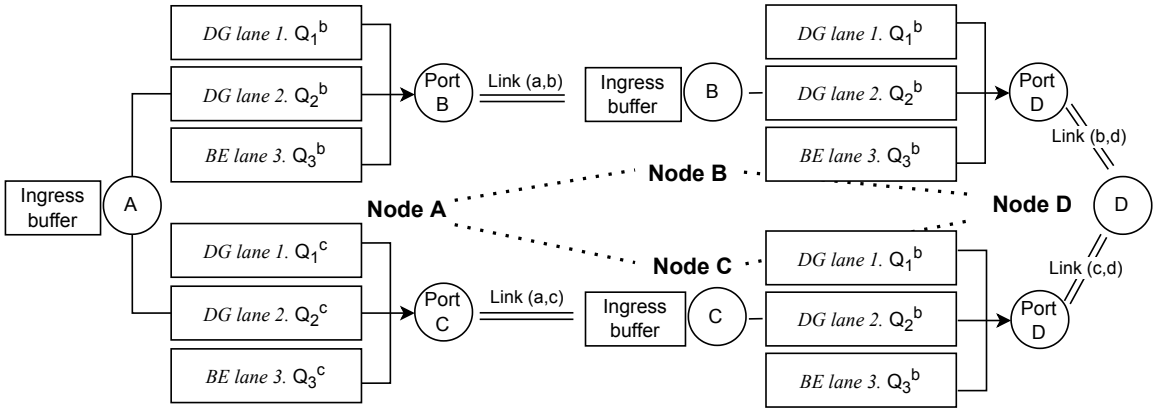


Figure 2.3: DSRP-enabled network with finite buffers

n nodes and an average number of k neighbor nodes. The time complexity of DSRP increases to $O(kn \log n)$, while the time complexity of a single-path routing protocol is $O(n \log n)$. The space complexity for maintaining a forwarding table at each router increases from n (for single-path routing protocol) to kn.

We numerically analyze the gain achieved by DSRP using a simple 2×2 grid topology as shown in Figure 2.3, where the link rate is fixed at 20 Mbps. A source application deployed in node A sends 1000 DS packets to the destination in node D with a constant sending rate. We increase the sending rate by 5 Mbps from 5 Mbps at each simulation. We compare DSRP with OSPF and show their throughput performance variations under different sending rates. As shown in Figure 2.4, the blue and red curves show the goodput performance of DSRP and OSPF, respectively, considering packet delay requirements (i.e., packets exceeding delay budget are ignored. For simplicity, we refer to it as goodput.). The black dashed curve indicates the total throughput achieved by OSPF without considering delay requirements, and the magenta curve indicates the sending rate of the application.

As shown in Figure 2.4, both DSRP and OSPF achieve high throughput when the sending rate is lower than the link capacity (i.e., 20 Mbps). Their performances diverge from 15 Mbps when the sending rate approaches the link capacity. OSPF in particular only achieves a total throughput of around 15 Mbps and a low goodput of less than 5 Mbps when the sending rate exceeds the link capacity. On the other

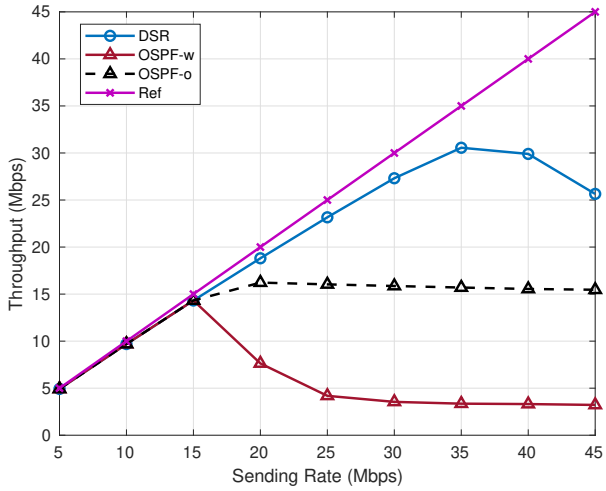


Figure 2.4: Numerical analysis of DSRP throughput improvements.

hand, DSRP maintains an increasingly high goodput performance and only starts to decrease when the sending rate exceeds 40 Mbps (2 times the link capacity). The performance gain achieved by DSRP is because both paths ($A \rightarrow B \rightarrow D$ and $A \rightarrow C \rightarrow D$) are used for packet transmission. Specifically, the scheduler at node A selects a different path if the original path is congested. However, when the sending rate approaches twice the link capacity, both paths become congested and packets will be dropped directly, causing goodput performance degradation.

In summary, DSRP adopts priority queues to provide DiffServ while using per-hop delay upper bound maintained by each router to explore multiple routing paths with various delay guarantees. In this context, a rich set of forward decisions including routing and queuing decisions at each hop can be obtained, which allows huge scheduling space to accommodate bursty traffic via load balancing. Therefore, an efficient routing and scheduling algorithm to determine the optimal forward decisions is crucial in the proposed DSRP network.

2.3.2 Network Model

We consider a network that is modeled as a graph $\mathcal{G} =< \mathcal{N}, \mathcal{L} >$, where $\mathcal{N}$ is the set of routers (we use node and router exchangeable in the rest of the chapter) and

$\mathcal{L}$ is the set of links. Let N and L be the number of nodes and links in the network, respectively. Denote R as the link rate and the link rate between two neighboring nodes i and j is denoted by $R_{i,j}$. Denote $\mathcal{N}_i$ as the set of neighbors of node i with the size N_i. For simplicity, we assume each node is connected with its neighbor node via a specific port and the port ID is the same as the neighbor ID. For example, in Figure 2.3, node A is connected to node B via port B at node A.

Let $\mathcal{K}$ be the set of priority queues deployed at each port and has the size of K. The buffer size of each queue is denoted by B_k, $k \in \mathcal{K}$ which guarantees a delay upper bound denoted by T_k, and $T_1 < T_2 < \cdots < T_K$. In addition, we assume each DS packet carries a tuple of information denoted by $\mathcal{C} = \{T_{budget}, T_{start}\}$ in its header, where T_{budget} is the delay budget and T_{start} is the set-off time at the source node[5]. The routers in DSRP network select a route and a corresponding queue for each packet. We couple the route and queue selection and refer to it as a forward decision. The forward decision for each DS packet is determined based on $\mathcal{C}$ at the ingress buffer and an NDS packet will be directly injected into the BE queue. Let $\mathcal{H}_i$ be the complete forward decision set at node i, then $\mathcal{H}_i = \{\mathcal{H}_{i,1}, \cdots, \mathcal{H}_{i,j}, \cdots, \mathcal{H}_{i,N_i}\}$, where $\mathcal{H}_{i,j}$ is the forward decision set at port j of node i. The size of $\mathcal{H}_i$ is denoted by H_i and thus, there are at most $(N_i - 1) \times K$ forwarding decisions at each node excluding the ingress port. Finally, let $h_{i,j}^k \in \mathcal{H}_i$ be the forward decision for each packet in node i, which indicates that the packet should be enqueued to queue k at port j.

2.3.3 Weighted-Round-Robin Scheduling Model

The WRR scheduling strategy is adopted to guarantee the per-hop delay upper bound of each DG queue and avoid low-priority queue starvation. Let T_{WRR} be the total service time of each round-robin period. Each queue is assigned with a portion of the service time denoted by w_k, where $w_1 > \cdots > w_K$ and $\sum_{k=1}^{K} w_k = 1$. The delay upper bound guaranteed by each queue can be expressed by $T_k = \frac{B_k}{w_k R},\ k \in \{1, 2, \cdots, K\}$.

[5]Here we assume all routers are synchronized using time-synchronization protocols, e.g., the IEEE 802.1AS standard [26].

We further assume that the WRR scheduler at each port switches to the next queue to serve if the current queue is empty so that no time slot is wasted.

2.3.4 Queuing Model

We consider a discrete-time system where packet arrivals and transmissions occur at the beginning of each normalized time slot. Arriving packets are backlogged at the corresponding queues before transmission. Let $\lambda_{i,j}^k[t]$ be the number of packets injected to queue k at port j of node i within time slot t. The number of packets dequeued from queue k to node j is denoted by $\mu_{ij}^k[t],\ j \in \mathcal{N}_i$, where $\mu_{ij}^k[t] = 1$ if the packet is successfully transmitted and $\mu_{ij}^k[t] = 0$, otherwise. Denote $Q_{i,j}^k$ as the queue length of queue k at port j of node i. Then the queue dynamics of queue k can be written as

$$Q_{i,j}^k[t+1] = \max\big\{Q_{i,j}^k[t] - \mu_{i,j}^k[t], 0\big\} + \lambda_{i,j}^k[t]. \tag{2.1}$$

The average queue length of all DG queues at port j after one WRR scheduling period is given by

$$\bar{Q}_{i,j} = \frac{1}{K-1} \sum_{k=1}^{K-1} \sum_{\tau=1}^{T_{\text{WRR}}} Q_{i,j}^k[\tau], \tag{2.2}$$

and similarly, the average queue length of all DG queues of node i is written as,

$$\bar{Q}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \bar{Q}_{i,j}, \tag{2.3}$$

which is useful for reflecting the congestion status of node i and generating NI shared with neighbor nodes.

2.3.5 Problem Formulation

We use a simple 4-nodes network topology to explain our problem as shown in Figure 2.3, where all packets at the ingress buffer of node A have the same destination to node D. Each port has three queues including two DG queues and one BE queue.

There are two paths to reach node D from node A, i.e., $r_1 := \{A \rightarrow B \rightarrow D\}$ and $r_2 := \{A \rightarrow C \rightarrow D\}$, and we assume that the costs of both paths satisfy the delay requirement of the application, i.e., $\max\{\text{Cost}(r1), \text{Cost}(r2)\} \leq T_b$, where $\text{Cost}(r) = \sum_{h \in r} \text{GI}_h$ denote the routing cost (i.e. E2E delay) and GI_h is the minimum delay upper bound at each hop h along path r. In this context, the forward decision set at each port of node A has the size $H_{AB} = H_{AC} = 3$ and a total size $H_A = H_{AB} + H_{AC} = 6$.

However, due to queue dynamics, not all queues in $\mathcal{H}_A$ satisfy the per-hop delay budget. Moreover, the congestion status at the downstream nodes (i.e., queue lengths of node B and C) also varies over time. A congested downstream node can lead to long queuing delay and heavy packet losses. Therefore, it is crucial to consider both local queue dynamics and neighbor queue information in routing and scheduling. In addition, since the WRR scheduler is adopted, there exists a mutual impact on queue selection. For example, heavy usage of the high-priority queues results in long queue delays in the low-priority queues because more time slots are used to serve the high-priority queues in each WRR period. Similarly, a busy low-priority queue affects the queuing time of the DS packets in the high-priority queues by using up the entire service time in each period.

Based on the above analysis, we formulate the following routing and scheduling problem aiming at maximizing the overall network utility. In brief, the optimization problem focuses on finding a series of suitable forward decisions for each packet such that its E2E delay requirement is satisfied while achieving load balance among priority queues at each hop. Mathematically, the problem can be formulated as follows

$$\text{P0:} \max_{h_d, h_n \in \mathcal{H}} \sum_{t=0}^{T} g_d(P_d[t]; h_d) + g_n(P_n[t]; h_n), \tag{2.4a}$$

$$\text{s.t.} \quad \text{Delay}(P_d[t]) \leq T_{budget}, \ \forall t \in \{0, 1, \cdots, T\}, \tag{2.4b}$$

$$Q_{i,j}^k[t] \leq \alpha B_k, \ \forall i, j \in \mathcal{N}, \ \forall k \in \mathcal{K}, \tag{2.4c}$$

where $g_d(\cdot)$ and $g_n(\cdot)$ denote the network utility function for DS and NDS packets, respectively; $P_d[t]$ and $P_n[t]$ denote the DS and NDS packets received by the destination at time slot t; h_d and h_n denote the forward decisions made at each hop along the path for DS and NDS packets, respectively; Delay$(\cdot)$ measures the E2E delay of each DS packet received at the destination, which is calculated by $T_{receive} - T_{start}$; and $\alpha \in [0, 1]$ is the parameter to control the maximum queue length of each queue. We fix $\alpha = 1$ for simplicity. Eq. (4a) aims at maximizing the overall network utility for the source-destination connection pair. Constraint (2.4b) ensures all DS packets received by the destination node should have E2E delay less than their delay budgets while constraint (2.4c) is used to avoid bufferbloat. Note that a packet will be dropped in the ingress buffer if (2.4b) or (2.4c) is not satisfied.

2.4 Algorithm Design

2.4.1 Delay-guaranteed Scheduling and Routing Protocol

The forward decision set grows with the increase of priority queues and the size of the network causing higher time complexity. However, some forward decisions are detrimental to the network performance causing longer delay and loops [83]. To address this issue, DSRP adopts a filter algorithm to filter out the undesirable routes in the forward decision set and determines the optimal forward decision jointly considering local queue information and neighborhood congestion status.

Route Filter Algorithm

Let Ω_i be the complete set of routes to reach the destination from node i in the forwarding table. Upon receiving a DS packet in the ingress buffer, the node computes a threshold value, a threshold value $T_{thld} = \min\{T^*_{budget}, \text{Cost}^*(r)\}$ is set as the threshold value to filter out the undesirable routes, where $T^*_{budget} = T_{budget} - (T_{now} - T_{start})$

Algorithm 2 Delay-guaranteed Scheduling and Routing Protocol

1: **Input:** forward decision set $\mathcal{H}_i$ at node i
2: **Output:** Forward decision $h_{ij}^{k*} = (route, queue)$
3: Upon receiving a packet at the ingress buffer of node i
4: Read T_{budget}, and T_{start} from the packet and router
5: Compute residual delay budget:
$T^*_{budget} = T_{budget} - (T_{now} - T_{start})$
6: Set $T_{thld} = \min\{T^*_{budget}, \text{Cost}^*(r)\}$
7: (**Part 1: Route filter**)
8: **for** each $route$ in Ω_i **do**
9: Let $Cost = \text{Cost}(route)$;
10: **if** $Cost > T_{thld}$ **then**
11: remove $route$ from Ω_i
12: **end if**
13: **end for**
14: **if** $\Omega_i = \varnothing$ **then**
15: Drop packet
16: **end if**

is the residual delay budget of the packet and $\text{Cost}^*(r)$ is the routing cost computed in the previous hop[6]. If Ω_i becomes empty after filtering, i.e., there is no available path that satisfies the delay requirement, the packet is dropped directly in the ingress buffer. Hence, the size of $\mathcal{H}_i$ can be reduced to $H_i = (N'_i - 1) \times K$, where N'_i is the size of Ω_i after filtering. The complete algorithm is shown in Algorithm 2 from line 3 to line 16.

In summary, the route filter algorithm ensures all the remaining routes in the forward decision set have at least one queue (i.e., the highest priority queue) that guarantees the delay requirement of the packet. It is also worth noting that the filter also helps to avoid the loop problem in dynamic routing by ensuring the routing cost of each forward decision selected at each hop is in descending order while approaching the destination.

[6]The design rationale of T_{thld} is to ensure the remaining routing paths satisfy the delay requirement, i.e., less than T^*_{budget}, and are closer to the destination compared to the previous hop to avoid loop problem. The major cause of loop in the DSRP network is the imbalance between a large delay budget and path diversity.

Algorithm 2 Delay-guaranteed Scheduling and Routing Protocol

1: (**Part 2: Congestion-aware route selection**)
2: Obtain Ω_i
3: **for** route $r_j \in \Omega_i$ **do**
4: **if** $\gamma_j = 1$ **then**
5: Remove r_j from Ω_i
6: **end if**
7: **end for**
8: Compute $T_{hop}(j) = T_{hop}(j) = T^*_{budget} - \text{Cost}(r_j)$
9: **for** $h^k_{i,j} \in \mathcal{H}_i$ **do**
10: Compute $T^k_{i,j}[t] = (1 + \gamma_j) \cdot \frac{Q^k_{i,j}[t] \cdot \text{PktSize}}{w_k \cdot R}$
11: **if** $T^k_{i,j}[t] > T_{hop}(j)$ **then**
12: Remove $h^k_{i,j}$ from $\mathcal{H}_i$
13: **end if**
14: **end for**
15: Obtain $\hat{\mathcal{H}}_i$
16: **if** $\hat{\mathcal{H}}_i = \varnothing$ **then**
17: Drop packet
18: **else**
19: Run **DSROpt**($\mathbf{H}_i$)
20: **Return:** Routing decision $h^{k*}_{ij} = (\text{port } j, \text{queue } k)$.
21: **end if**
22: Enqueue packet to queue k at port j

Congestion-aware Route Selection Algorithm

Although the filtered forward decision set $\mathcal{H}_i$ satisfies the delay requirement, packets may be dropped due to network congestion in the downstream nodes. We address this issue by enabling queue information exchange among neighboring nodes. Inspired by the idea from backpressure routing algorithms [100] where the backlog gradient is used for packet scheduling, we develop a congestion-aware route selection algorithm as shown in Algorithm 2 Part 2 from line 2 to line 22.

The algorithm works based on the design of VQM which periodically exchanges local queue information with neighboring nodes. Specifically, the VQM at each node computes the average queue length using Eq. 2.3 to estimate the current queuing status. An indicator function γ based on the average queue length is designed as

follows

$$\gamma_j = \begin{cases} \frac{\bar{Q}_j^1}{\alpha \bar{B}_1} & \bar{Q}_j^1 < \alpha \bar{B}_1, \\ 1 & \bar{Q}_j^1 \geq \alpha \bar{B}_1, \end{cases} \quad j \in \mathcal{N}_i, \tag{2.5}$$

where $\bar{Q}_j^1$ and $\bar{B}_1$ refers to the average queue length and average buffer size of the all highest priority queues of the node[7]. Such design helps to deliver the congestion status of the downstream node to the upstream node. If node i receives a congestion signal $\gamma_j = 1$ from node j indicating heavy congestion, it removes the route to reach node j from Ω_i to alleviate the congestion of the downstream nodes. Otherwise, it is used as a control parameter for route evaluation.

Next, for each decision $h_{i,j}^k \in \mathcal{H}_i$, the router computes a weighted queue delay denoted by $T_{i,j}^k[t]$, where

$$T_{i,j}^k[t] = (1 + \gamma_j) \cdot \frac{Q_{i,j}^k[t] \times \text{PktSize}}{w_k R} \tag{2.6}$$

and the per-hop delay budget T_{hop} of each output port

$$T_{hop}(j) = T_{budget}^* - \text{Cost}(r_j),\ r_j \in \Omega_i\ j \in \mathcal{N}_i. \tag{2.7}$$

The weighted queue delay $T_{i,j}^k[t]$ of each forward decision $h_{i,j}^k \in \mathcal{H}_i$ is compared with $T_{hop}(j)$, and $h_{i,j}^k$ will be removed from $\mathcal{H}_i$ if $T_{i,j}^k[t] > T_{hop}(j)$, i.e., the per-hop delay budget is insufficient to support the corresponding forward decision. Note that the packet is dropped directly if the forward decision set is empty, i.e., no available decision exists. Finally, a fully simplified forward decision set $\hat{\mathcal{H}}_i,\ i \in \mathcal{N}$ that guarantees the E2E delay is obtained. Next, the router runs the proposed DSROpt scheduling algorithm to determine the optimal forward decision h_{ij}^{k*}.

[7]This is because the highest priority queue guarantees the minimum delay upper bound of each hop and is preserved for urgent packets by design (explained in Algorithm 3), its queue length is sufficient to reflect the congestion status of the corresponding node.

2.4.2 Network Utility Maximization

We aim to maximize the network utility as presented in P0. However, since the forward decisions are spatio-temporally correlated, i.e., the current decision is affected by decisions made in previous time slots and upstream nodes, and eventually affects the overall network utility. Therefore, it is difficult if not impossible to find the optimal scheduling policy at every node for every packet given the large scheduling space, stringent delay requirement, and network dynamics.

To this end, we transform the utility maximization problem and let each node computes the optimal forward decision for each packet. Specifically, at time slot t, we represent each forward decision $h_{i,j}^k \in \hat{\mathcal{H}}_i$ by a cost-reward tuple $(T_{i,j}^k[t], G_{i,j}^k[t])$, where $G_{i,j}^k[t]$ is the reward obtained for executing $h_{i,j}^k$. The infinite horizon reward per unit time cost ratio is then given by

$$\theta = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} G_{i,j}^k[t]}{\sum_{t=0}^{T} T_{i,j}^k[t]}, \tag{2.8}$$

assuming the limit exists. In this context, the overall network utility can be improved if θ_i at each node $i \in \mathcal{N}$ is maximized. Note that to avoid the divide-by-zero issue, we set $T_{i,j}^k[t] = 1$ if the queue is empty.

In other words, we decouple the end-to-end network utility maximization problem in P0 into a series of subproblems, where the reward per unit time cost ratio at each node is maximized. Therefore, P0 can be rewritten as follows

$$\text{P1:} \quad \max_{h_{i,j}^k \in \hat{\mathcal{H}}_i,\ i\in\mathcal{N}} \quad \theta_i = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} G_{i,j}^k[t]}{\sum_{t=0}^{T} T_{i,j}^k[t]}, \tag{2.9a}$$

$$\text{s.t.} \quad (T_{i,j}^k[t], G_{i,j}^k[t]) \in \mathcal{D}(\hat{\mathcal{H}}),\ t \in \{0, 1, \cdots, T\}, \tag{2.9b}$$

where $\mathcal{D}(\hat{\mathcal{H}})$ denotes the set of all possible cost-reward combinations in the forward decision set $\hat{\mathcal{H}}$. Since the forward decision set is only known when the packet's delay requirement is observed and the state of the system is renewed after executing each

forward decision, we identify each subproblem as a renewal optimization problem [96, 98]. However, due to the randomness of network dynamics and large space of $\mathcal{D}(\hat{\mathcal{H}})$, it is impractical to find the optimal solution to P1 [98]. To address this issue, we propose a heuristic algorithm that maximizes (2.9a) by learning from past forward decisions.

Delay laxity-based reward function

We first discuss the design of our reward function which will be used in our proposed scheduling algorithm. We focus on the scheduling of DS packets, as the NDS packets are directly forwarded to the BE queue. Since there are multiple priority queues for transmitting DS packets, it is unfair to use the same reward function to measure the output gain of successful transmissions of DS packets using different priority queues[8]. In this regard, we define a metric called *profit* to measure the true return of using each queue. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$ be the set of profit to use each queue from high priority to low priority and $P_1 < P_2 < \ldots < P_{K-1}$[9]. Furthermore, we define a delay laxity as denoted by

$$L_{ij}^k|_{h_{ij}^k} = T_{hop}(j) - T_{i,j}^k, \quad h_{ij}^k \in \hat{\mathcal{H}}_i \tag{2.10}$$

as the residual per-hop delay budget, which is helpful in measuring the urgency of each DS packet [112]. Finally, we define the reward function for successfully transmitting the DS packets of choosing decision h_{ij}^k from $\hat{\mathcal{H}}_i$ as follows

$$G_{i,j}^k|_{h_{ij}^k} = P_k \cdot L_{ij}^k = P_k \cdot (T_{hop}(j) - T_{i,j}^k), \ h_{ij}^k \in \hat{\mathcal{H}}_i. \tag{2.11}$$

The goal of such design is to first use the lower priority queues when available while preserving the higher priority queues for the urgent packets yet to come.

[8]Otherwise, the high priority queues are more likely to be selected as they provide smaller delay upper bounds.

[9]For simplicity, we set the profit of each DG queue to be inversely proportional to the weight assigned to them, i.e., $P_k = 1/w_k$.

Algorithm 3 DSROpt scheduling algorithm

Input: forward decision set $\hat{\mathcal{H}}_i$, η
Output: forward decision $h_{i,j}^k$
Initialize $\phi, \phi' = 0$, $\theta[t] \in [\theta_{\min}, \theta_{\max}]$,
for each decision $h_{ij}^k \in \hat{\mathcal{H}}_i$ at node i **do**
  Compute $T_{i,j}^k[t]$ and $G_{i,j}^k[t]$ using (2.6) and (2.11)
  Let $\phi = G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t]$
  if $\phi \geq \phi'$ **then**
    $\phi' = \phi, \text{port} = j, \text{queue} = k$
  end if
end for
Update $\theta[t+1] = \left[\theta[t] + \eta(G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t])\right]_{\theta_{\min}}^{\theta_{\max}}$
Return: decision $h_{i,j}^k = (\text{port } j, \text{queue } k)$

Renewal optimization-based scheduling algorithm

Each router in the DSRP network is essentially a renewal system that updates its state, i.e., queue status, after executing each forward decision. But due to network randomness and lack of packet knowledge, it is impractical to find the optimal scheduling strategy that maximizes (2.9a). To this end, we develop a heuristic scheduling algorithm DSROpt based on renewal optimization as shown in Algorithm 3.

Although packet arrival is unknown, the cost-reward pairs of all possible combinations are bounded. For example, the output gain for each DS packet is bounded by

$$G_{i,j}^k|_{h_{ij}^k} \in [G_{\min}, G_{\max}] = [0, P_K \cdot \max\{T_{hop}\}], \ h_{ij}^k \in \hat{\mathcal{H}}_i.$$

Similarly, the time cost $T_{i,j}^k$ is bounded by $[T_{i,j}^1, T_{i,j}^K]$, where $T_{i,j}^1$ and $T_{i,j}^K$ are the delay upper bounds maintained by the highest priority queue and the lowest priority queue, respectively.

Assume the tuple $\{T_{\min}, T_{\max}, G_{\min}, G_{\max}\}$ is fixed at each node, where

$$T_{\min} = \min\left\{T_{i,j}^k | i, j \in \mathcal{N}, \ k \in \mathcal{K}\right\},$$

$$T_{\max} = \max\left\{T_{i,j}^k | i, j \in \mathcal{N}, \ k \in \mathcal{K}\right\},$$

$$G_{\min} = \min\left\{G_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\},$$

$$G_{\max} = \max\left\{G_{i,j}^k | i,j \in \mathcal{N},\ k \in \mathcal{K}\right\}.$$

Let

$$\theta_{\min} = \min\left\{\frac{G_{\min}}{T_{\max}}, \frac{G_{\min}}{T_{\min}}\right\},\quad \theta_{\max} = \max\left\{\frac{G_{\max}}{T_{\max}}, \frac{G_{\max}}{T_{\min}}\right\}$$

denote the minimum and maximum reward per unit time cost ratio, respectively. Then the optimal θ^* satisfies the condition $\theta_{\min} \leq \theta^* \leq \theta_{\max}$ if exists.

Note that always choosing the decision which yields the maximum reward does not necessarily maximize θ over time given the temporal correlations between forward decisions. To solve the optimization problem in P1, an iterative approach proposed in [98] is adopted, where θ^* can be approached by constantly selecting the cost-reward pair that maximizes the reward gradient $\phi = G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t]$. In other words, the router determines the optimal forward decision by solving the following problem

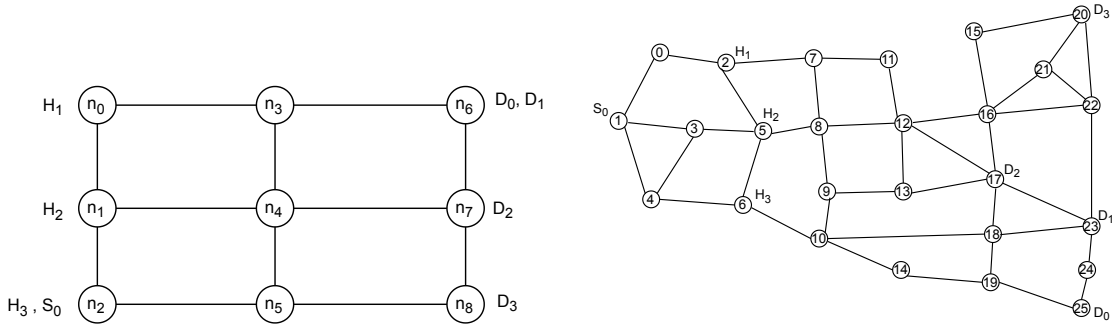
$$\text{P2:} \quad \max_{h_{i,j}^k \in \hat{\mathcal{H}}} \ G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t], \tag{2.12a}$$

$$\text{s.t.} \quad (T_{i,j}^k[t], G_{i,j}^k[t]) \in \mathcal{D}(\hat{\mathcal{H}}). \tag{2.12b}$$

Here $\theta[t]$ is updated after each decision

$$\theta[t+1] = \left[\theta[t] + \eta[t](G_{i,j}^k[t] - \theta[t]T_{i,j}^k[t])\right]_{\theta_{\min}}^{\theta_{\max}}, \tag{2.13}$$

where η is the step size of each iteration and $[x]_{\theta_{\min}}^{\theta_{\max}}$ denotes the normalization of x within the range of $[\theta_{\min}, \theta_{\max}]$. The convergence of the update rule (2.12) is discussed in the Appendix. Such an opportunistic online learning approach is effective because the historical decisions are included in the optimization objective. The complete scheduling algorithm is summarized in Algorithm 3. To be more specific, each node $i \in \mathcal{N}$ initializes the reward per time cost ratio $\theta[t]$ by randomly selecting a value from the range $[\theta_{\min}, \theta_{\max}]$ at $t = 0$. To determine the optimal forward decision for a DS packet received at time slot t, the node first computes the weighted queue delay $T_{i,j}^k[t]$



(a) 3×3 grid topology. (b) U.S. backbone network.

Figure 2.5: Network topologies used in the experiments.

and output gain $G_{i,j}^k[t]$ for each available forward decision $h_{i,j}^k \in \hat{\mathcal{H}}_i$ of the packet. Then, it selects the forward decision that solves P2 and updates $\theta[t]$ according to (2.13).

2.5 Performance evaluation

In the experiment, we implemented DSRP using the ns-3 network simulator [113]. Three priority queues including two DG queues and one BE queue are deployed at each port of each router. Note that the model can be easily extended with more priority queues to achieve a finer-granularity performance.

We compare the performance of DSROpt scheduling algorithm (denoted by DSR in our simulation results) with three BP-based scheduling algorithms such as the finite-buffer max-weight algorithm [71] (denoted by MAX), sojourn-time-based BP algorithm [52] (denoted by TBP), MW+BP algorithm developed in [132] (denoted by MBP), and one greedy algorithm denoted by GRD which always selects the route/queue with the highest reward calculated by Eq. (2.11). All algorithms are implemented based on the proposed DSRP network architecture with priority queues. In addition, we use a single-queue network that adopts OSPF routing protocol (denoted by BEF) for packet delivery as the benchmark in our experiment. Since we assume a static network topology and link cost in our experiment, OSPF provides

Table 2.2: Network configurations

Parameters	DG queue 1	DG queue 2	BE queue
Weight	0.5	0.3	0.2
Buffer size (packet)	6	17	100
Network type	Source	Destination	
Grid	n_2, n_0, n_1	n_6, n_7, n_8	
Mesh	n_0, n_2, n_5, n_6	$n_{17}, n_{20}, n_{23}, n_{25}$	

comparable performance as a centralized routing algorithm in static networks.

2.5.1 Network Configurations

We consider two types of network topology in the experiments, i.e., a 3×3 grid network and a 26-nodes mesh network as shown in Figure 2.5. Every router in both networks has the same configurations as summarized in Table. 2.2. The buffer size and weight assigned to each queue are $1200, 3400$, and 20K bytes, and $0.5, 0.3$, and 0.2, respectively. The link rate of all links in both networks is fixed at 20 Mbps unless mentioned otherwise. All packets have the same size of 200 bytes. Therefore, the DG queues guarantee a per-hop delay upper bound of 2 ms and 10 ms, respectively. The propagation delay of each link is randomly selected from a $[3, 5]$ ms range.

We use the (Src, Dst) pair to indicate the source-destination connection in our experiments. In the grid network, (n_2, n_6) is used as our target pair, i.e., (S_0, D_0) as shown in Figure 2.5, for analysis while three other pairs (n_0, n_6), (n_1, n_7), and (n_2, n_8) are set to simulate the background traffic in the network. In the mesh network, (n_0, n_{25}) is the target pair. Three pairs (n_2, n_{20}), (n_5, n_{17}), and (n_6, n_{23}) are used for simulating background traffics. Note that in our experiments, we only evaluate the performance of the target pairs.

2.5.2 Performance Metrics

Several performance metrics are defined to measure the performance of each algorithm: (1) the E2E delay of each DS packet, (2) the average E2E delay of all DS packets and their standard deviation, and (3) the goodput calculated by $\frac{\text{total received bits}}{\text{total Tx time}}$.

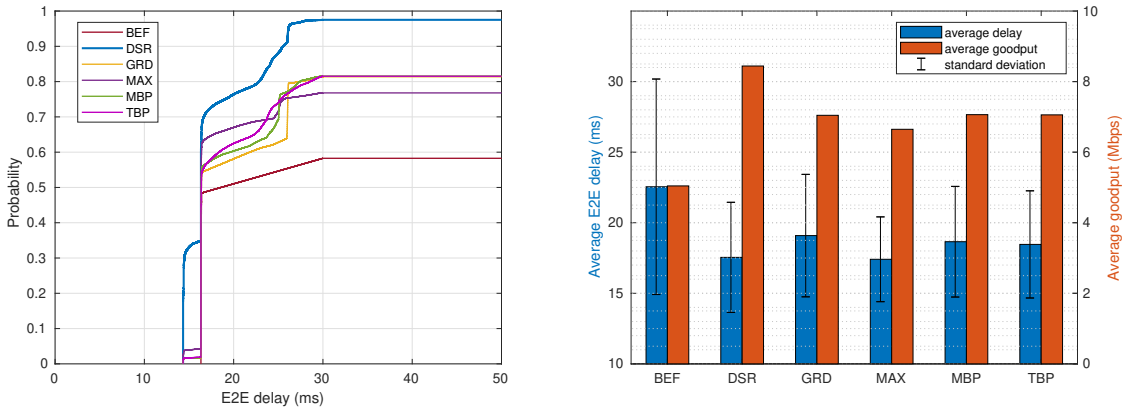
Table 2.3: Experiment settings

Impact of	Target pair	Background pair
Bursty traffic	Base rate: 5 Mbps Bursty range: 1∼15 Mbps Base rate: 1 Gbps Bursty range: 0∼2.5 Gbps	Base rate: 10 Mbps Bursty range: 1∼5 Mbps Base rate: 1 Gbps Bursty range: 0∼0.5 Gbps
Traffic density	Base rate: 10 Mbps	Base rate: 5 Mbps (DS) Increase by 1 Mbps (NDS) Increase by 2 Mbps
Delay budget	Base rate: 10 Mbps Delay budget reduce by 2 ms per epoch	(DS) Base rate: 5 Mbps Bursty range: 1∼15Mbps (NDS) Base rate: 10 Mbps Bursty range: 1∼5 Mbps
NDS traffics	Base rate: 5 Mbps Increase 2 Mbps per epoch	(DS) Base rate: 5 Mbps (NDS) Base rate: 10 Mbps

We also define the performance gain as $a_1 N_{\mathrm{DS}} + a_2 N_{\mathrm{NDS}}$, where a_1 and a_2 $(a_1 > a_2)$ are the utility gain for successfully receiving DS packet and NDS packet at the destination. In our experiments, we fix $a_1 = 3, a_2 = 1$. Note that all DS packets that exceed their delay budget are discarded in the network or at the receiver side, i.e., are not considered in performance evaluations.

2.5.3 Simulation results and analysis

We designed four experiments to evaluate our proposed delay-guaranteed solution. Specifically, we tested its delay and goodput performance under bursty traffic, different traffic volumes, various delay requirements, and co-existence with NDS traffic. Each experiment consisted of several simulation rounds with different settings, called epochs. All experiments used UDP traffic and their settings are summarized in Table 2.3. In addition, we select two representative algorithms, TBP and BEF, to show their standard deviations in end-to-end delay. TBP is selected because it leverages the local queuing delay to make scheduling decisions and it performs the second best in our experiments.



(a) CDF of per-packet E2E delay.

(b) Average QoS performances.

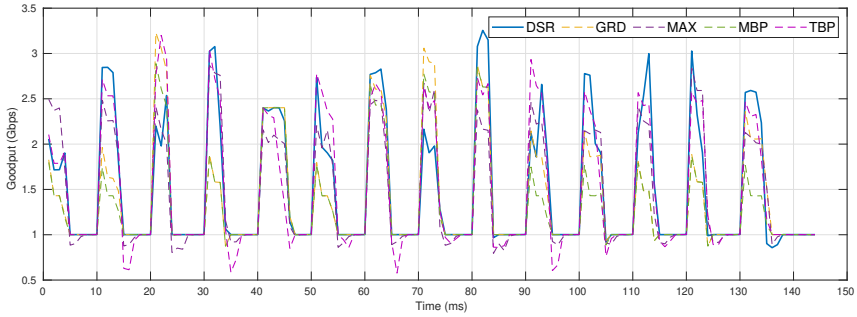
Figure 2.6: Comparisons under bursty traffic

Impact of bursty traffic

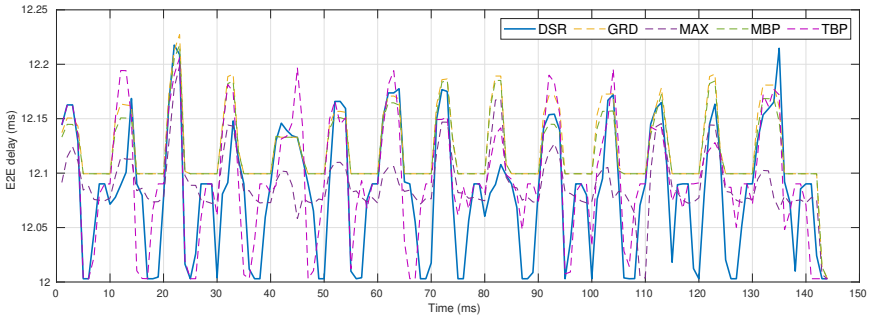
In this experiment, we verify the performance of DSROpt scheduling algorithm under bursty traffic using the 3×3 grid topology. Both target and background traffic pairs transmitted DS packets with bursty sending rates. The target source node transmitted 50000 packets at a base rate of 5 Mbps and an additional bursty rate of $1 \sim 15$ Mbps. The background traffic pairs had a base rate of 10 Mbps and an additional bursty rate of $1 \sim 5$ Mbps. All DS packets required a 30 ms end-to-end delay performance.

The simulation results are presented in Figure 2.6, where our algorithm achieves the best delay and goodput performance compared with other algorithms. As shown in Figure 2.6(a), DSROpt guarantees the most DS packets delivery up to 98% within 30 ms. GRD, MBP, and TBP show similar performances guaranteeing around 82% of the DS packets while MAX achieves 76% of E2E delay guarantee. BEF only guarantees 59% of the packets due to severe packet drop caused by congestion and bufferbloat.

Without loss of generality, we also compare the average delay and goodput performances of all algorithms as shown in Figure 2.6(b). As expected, DSROpt shows the highest goodput up to 8.2 Mbps while GRD, MBP, and TBP give similar good-



(a) Goodput performance.

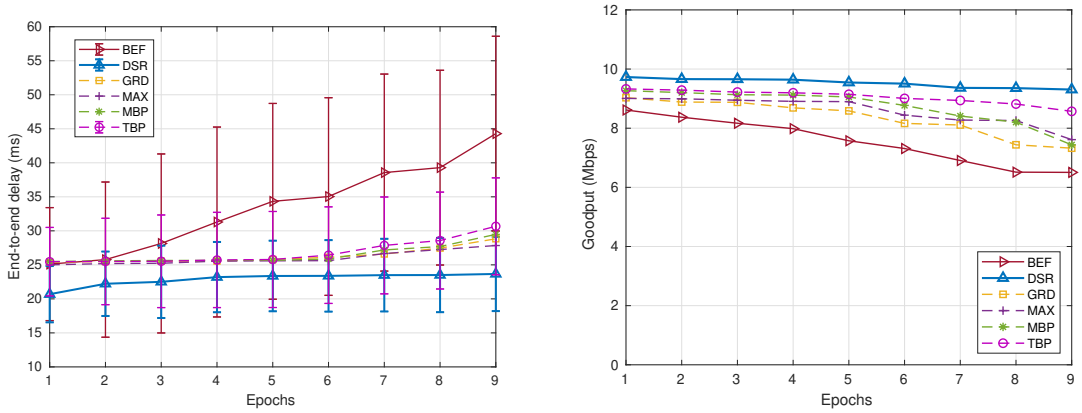


(b) E2E delay performance.

Figure 2.7: QoS performances with high link rate

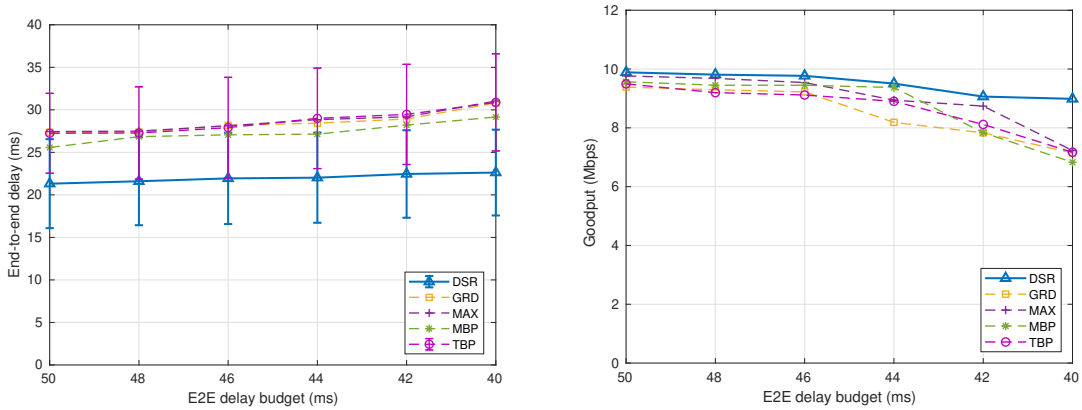
put around 7.2 Mbps. MAX shows a worse goodput performance of 6.7 Mbps. BEF gives the worst goodput performance of 5.2 Mbps. As for the average E2E delay, the BP-based scheduling algorithms achieve similar delay performances, among which MAX shows slightly better performance as it achieves load balance using the queue-length gradient. Overall, the proposed priority queue-based network outperforms the single-queue network because the upstream nodes are able to change routes for the DS packets. DSROpt outperforms other algorithms thanks to its congestion-aware design which helps to avoid congested paths to improve goodput.

In addition, we verify the scalability of our network architecture and DSROpt scheduling algorithm using a high link rate network setting. Each link has a rate of 2 Gbps. The target source application transmits DS data packets with a bursty sending rate ranging from $1.0 \sim 3.5$ Gbps and the background traffics have a sending rate of $1 \sim 1.5$ Gbps. The goodput and delay performances of DSROpt and other



(a) Average delay performance. (b) Goodput performance.

Figure 2.8: QoS performance comparison using mesh network



(a) Average E2E delay. (b) Goodput.

Figure 2.9: Comparisons under different delay budgets.

algorithms are shown in Figure 2.7. The results show that DSROpt has good scalability and is able to maintain high goodput and low E2E delay in high data rate network settings.

Impact of traffic density

In this experiment, we use the mesh network topology to verify the performance of DSROpt under various traffic densities. In the target pair, the source node transmits 10000 DS packets with a delay budget of 45 ms to the receiver at a fixed sending

rate of 10 Mbps in each epoch. As for the background pairs, both DS and NDS applications are deployed. The sending rate of the DS and NDS application increases by 1 Mbps and 2 Mbps at each epoch from 5 Mbps, respectively.

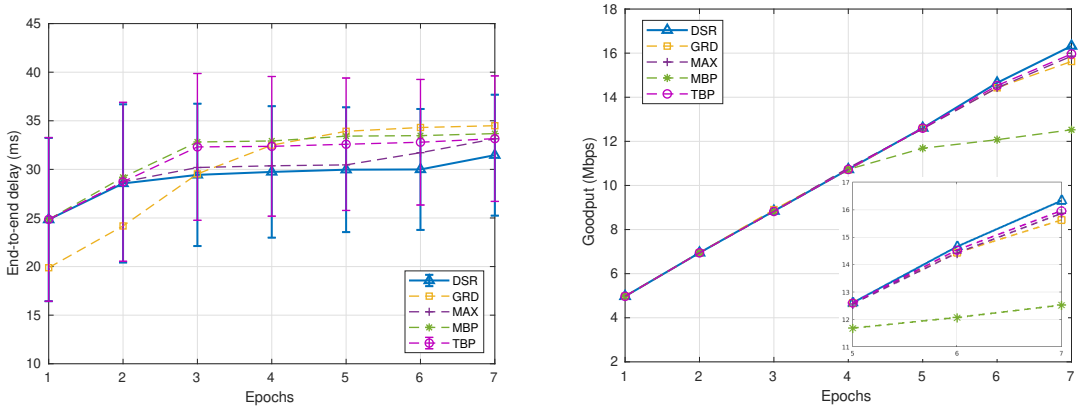
Figure 2.8(a) shows the delay performance. The E2E delay of BEF increases quickly because all traffic uses the same queue for packet transmission and OSPF always chooses the shortest path for packets routing, which can easily result in network congestion when the packet arrival rate exceeds the link capacity, leading to a long queue delay and bufferbloat. This also accounts for the large end-to-end delay variance in each epoch. BP-based algorithms (GRD, MAX, TBP, and MBP) show similar performances, degrading from the 5th epoch when the aggregated sending rate exceeds link bandwidth. DSROpt performs best throughout the experiment, reducing end-to-end delay by up to 22.5% compared to BP-based scheduling algorithms.

Figure 2.8(b) shows the goodput performance. BEF performs the worst due to heavy congestion and packet drops as traffic density increases. MAX, GRD, and MBP show similar patterns as the sending rate grows, with GRD experiencing more rapid performance degradation due to congestion. TBP has better goodput performance with slower degradation because it considers each DS packet's sojourn time and schedules urgent packets to higher priority queues, reducing delay outage drops. On the other hand, DSROpt maintains the highest goodput throughout the experiment and is only reduced by 0.8 Mbps until the last epoch. Overall, DSROpt achieves a goodput increase of up to 26.4% over BP-based scheduling algorithms.

Impact of delay requirements

Furthermore, We show the impact of the delay requirements using the mesh network topology. Similar to the previous settings, the target pair transmitted 20000 DS packets at a fixed rate of 10 Mbps while the background pairs transmitted 10000 mixed DS and NDS packets at a varying rate of $[10, 30]$ Mbps in each epoch. The delay budget for the target application decreased from 50 ms to 40[10]. The simulation

[10]Note that the minimum E2E cost that can be guaranteed for the target pair is 40 ms.



(a) DS application: Average delay.

(b) DS application: Goodput.

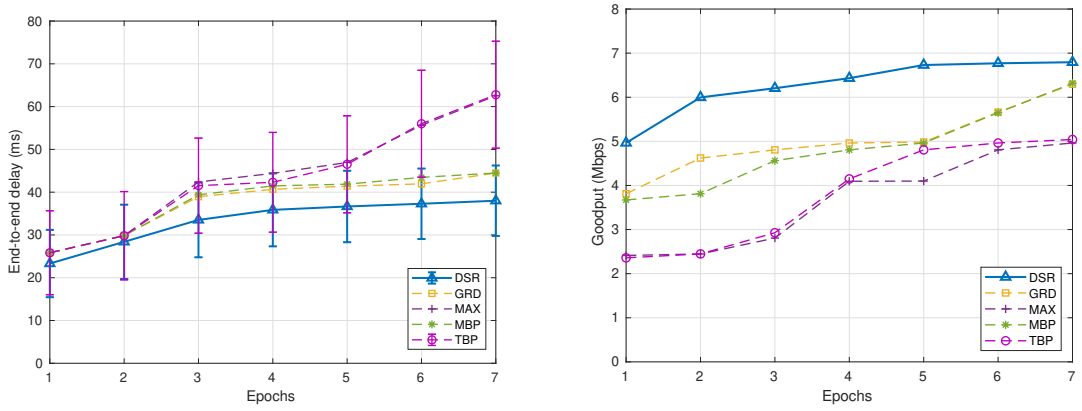
Figure 2.10: QoS performance comparisons on DiffServ property for DS applications.

results are shown in Figure 2.9.

Overall, GRD, MAX, TBP, and MBP show similar delay and goodput performance. All experience rapid goodput degradation when the delay budget is less than 44 ms, approaching the target pair's minimum delay cost to reach the destination. DSROpt maintains a good performance, reducing delay by up to 28.1% and improving goodput by up to 47.5% compared to other BP-based scheduling algorithms. It maintains a goodput of 9 Mbps when the delay budget is 40 ms while BP-based algorithms degrade to less than 7 Mbps. This is because DSROpt considers each packet's delay requirement and temporal correlations among forward decisions for scheduling. On the other hand, BP algorithms aim to minimize per-hop time cost in scheduling packets, frequently occupying high-priority queues and leading to goodput degradation when the delay budget is small.

Differentiated service

The DSRP network architecture can provide DiffServ to both DS and NDS traffic simultaneously and achieves the highest performance gain using DSROpt. We verify this property using the grid topology shown in Figure 2.5. Two applications, one DS with a delay requirement of 35 ms and one NDS, are deployed at the target and



(a) NDS application: Average delay .

(b) NDS application: Goodput.

Figure 2.11: QoS performance comparisons on DiffServ property for NDS applications.

background pairs. In each epoch, we increase both applications' sending rate by 2 Mbps from 5 Mbps for the target pair. The background pairs' sending rates are fixed at 5 Mbps for the DS application and 10 Mbps for the NDS application. We evaluate both applications' QoS metrics for the target pair and only compare DSROpt with the BP-based scheduling algorithms.

The QoS performances of DS applications are presented in Figure 2.10. As shown in Figure 2.10(a), the average E2E delay achieved by all algorithms increases in each epoch due to the longer queuing delay. GRD achieves the least delay when the network is less congested but quickly grows to the highest as the sending rate increases due to its greedy exploration strategy causing congestion and long queuing delays. DSROpt performs best when the network becomes more congested because it considers both downstream and current node congestion when making forward decisions. As for the goodput performance shown in Figure 2.10(b), all algorithms have similar performances when the network is underloaded. BP-based algorithms gradually diverge when network traffic exceeds link capacity from the 4th epoch. MBP degrades quickly from the 5th epoch when the target pair's aggregated sending rate exceeds 20 Mbps and the rest saturate from the 6th epoch. DSROpt shows an

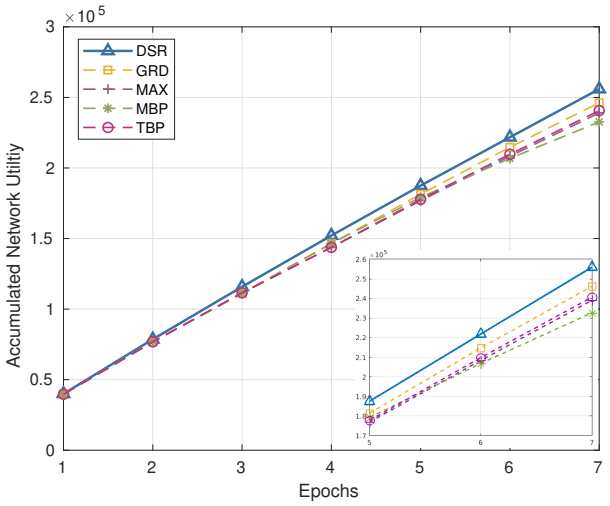


Figure 2.12: Performance gain achieved by different algorithms.

approximately linear goodput increase, reaching up to 16.2 Mbps when the DS packet sending rate is 17 Mbps, with TBP achieving the second-best goodput of 16 Mbps.

In terms of NDS applications, the QoS performance presents a different pattern as shown in Figure 2.11. Two groups, GRD and MBP, and MAX and TBP, show similar performances in delay and goodput, respectively. All algorithms show low goodput and increasingly high end-to-end delay as the sending rate grows due to a single queue and shortest path being used for NDS packet transmission. The MAX and TBP group, in particular, sees a sharp delay increase when the sending rate increases because they heavily exploit optimal queues/routes for DS packet transmission while starving low-priority queues transmitting NDS packets. On the other hand, DSROpt achieves the best performances in both E2E delay and goodput compared to the BP-based scheduling algorithms because it considers the mutual impact of forward decisions by maximizing the long-term reward per unit time cost ratio at each node, maintaining good performance for both DS and NDS applications.

We also compared the network performance gain achieved by different scheduling algorithms as shown in Figure 2.12. DSROpt achieves the best performance, 10.2% higher than MBP. Although the pattern varies by tuning the utility gain (i.e., a_1 and a_2) of DS and NDS packets, DSROpt always performs best. We can conclude that

DSRP can provide DiffServ to different applications with different delay requirements.

2.6 Conclusion

In this chapter, we investigate a novel distributed, congestion-aware, and delay-guaranteed solution for time-sensitive applications named DSRP. A DSRP-enabled network provides multiple routes to reach the destination by design, therefore, an efficient routing algorithm that achieves high network utility while guaranteeing the E2E delay is desired. To this end, we propose DSROpt, a novel scheduling algorithm that selects routes and queues for each packet based on the delay requirement, downstream congestion status, and local queue information. To verify the performance of our solution, we implemented a prototype on ns-3 and compared it with the state-of-the-art via simulations. The results show that DSROpt guarantees the E2E delay requirements while achieving the highest goodput and network utility in various experiments.

Chapter 3

Downlink Scheduler for Delay Guaranteed Services Using Deep Reinforcement Learning

In Chapter 2, we focused on the scheduling and routing problem in wired data networks where we proposed a novel network architecture using priority queues to guarantee various end-to-end delay requirements. However, the proposed solution may not be effective and cause delay outage in the last hop especially when packets arrive at the end of the queue with a small remaining delay budget. In addition, most time-sensitive applications nowadays are implemented in a wireless network setting and its unstable channel condition feature adds more challenges to supporting delay-guaranteed service. In this chapter, we focus on the downlink scheduling problem and aim to address the aforementioned issues. We explore the possibility of further improving network utility by considering a novel delay-aware selective scheduling algorithm.

Table 3.1 summarizes the notations and definitions used frequently in this chapter.

Table 3.1: Notations and definitions

Symbol	Definitions
$\mathcal{K}$,	Sets of packets
$\mathcal{U}$	Sets of users associated the BS
$\mathbf{A}(t)$	Packet admission matrix
$S_u(t)$	Control variable of serving user u at slot t
$\boldsymbol{\eta}(t)$	Transmission status of packets
$\eta_{u,k}(t)$	Variable of transmitting packet k to user u at slot t
$\mathbf{C}(t)$	Channel condition vector
$C_u(t)$	Channel condition of downlink u at slot t
$1_u(t)$	Value of successful transmission of downlink u at slot t
$\mathbf{Q}(t)$	Set of packets backlogged in each queue
$Q_u(t)$	Backlogged packets of queue u at slot t
$x_{u,k}(t)$	Sojourn time of packet k in queue u
Th_H	Maximum tolerable delay of packet $k \in \mathcal{H}$
$D_{u,k}(t)$	Queuing state of packet k in queue u
$L_{u,k}(t)$	Delay budget laxity of packet k in queue u
ω_H	Gain weights of packets with high delay requirements
ρ	Discount factor
$G(t)$	Achievable gain of the system at slot t
$\bar{A}_u$	Average packets admitted to queue u over all slots
λ_i^n	Weight of objective i in the n-th subproblem

3.1 Introduction

The resource scheduling problem of wireless networks has been extensively studied in various contexts, where throughput and delay are the main performance index [31, 70, 99]. For the case when the traffic is inside the capacity region, the throughput is equal to the arrival rate and then the throughput maximization problem reduces to a network stability problem. The network can be stabilized by a max-weight policy that schedules links per time to maximize a weighted sum of transmission rates, where the weights are queue backlogs [45]. This is typically shown by the Lyapunov drift

theory [40]. For a general case when the traffic is either inside or outside of the capacity region, the max-weight policy can be combined with a flow control policy to jointly maximize the throughput and stabilize the network [9,72,85]. In [93] and [53], a delay-based Lyapunov function was proposed, where the delay of the head-of-queue packet is served as a weight of the max-weight decision. Although there are few works that use delay-based scheduling to solve joint stability and utility optimization problems, it is not suitable for applications with stringent delay requirements.

In this chapter, we propose a delay-aware scheduling policy to guarantee the delay of time critical packets. Specifically, we consider a single-hop downlink network following a similar network assumption in Chapter 2 where packets are prioritized in terms of delay requirements the delay information is carried by each packet. To describe the utility of different classes of packets, we define an output gain function where packets with high delay requirements yield high output gain. Although scheduling high delay requirement packets can obtain high output gain, it may result in packets in lower classes being dropped unnecessarily due to excessive delay. Therefore, a delay-laxity concept is introduced where packets with the least laxity are prioritized. In this context, we formulate a multi-objective optimization problem (MOOP) aiming to minimize the average queue length while maximizing the average output gain under the constraints of guaranteeing per-packet delay and achieving fairness among users.

Although dynamic programming methods can solve this problem, if the environment changes over time, the solution has to be recalculated which may take a similar amount of time as the initial solution. To this end, we provide a new framework for solving this MOOP through deep reinforcement learning. We decompose it into a set of scalar subproblems using a weighted sum approach and solve each subproblem cooperatively with other subproblems through a neighborhood-based parameter transfer strategy. We then model each subproblem as a partially observable Markov Decision Process (POMDP) and resort to a double deep Q network (DDQN)-based scheduling algorithm to learn. Moreover, our proposed DDQN-based algorithm can

address the curse of dimensionality with large state and action spaces and reduce Q-value overestimation.

The rest of the chapter is organized as follows. We discuss the related works in Section 3.2. In Section 3.3, we explain our system model and formulate the network optimization problem with detailed analysis. In Section 3.4 and Section 3.5, we discuss the proposed DRL-based framework and DDQN-based algorithm, respectively. Extensive simulations are designed to verify the performance of our model and algorithm in Section 3.6. Finally, we conclude our work in Section 3.7.

3.2 Related Works

There have been many studies on wireless resource management using stochastic optimization. In particular, the theory of Lyapunov drift and optimization has been used to ensure network stability and utility optimization [65]. When the arrival rate is within the capacity region, the throughput optimization problem reduces to the network stability problem and then the Lyapunov drift technique has been employed to stabilize the network by greedily minimizing the Lyapunov drift every time slot. The max-weight or backpressure algorithm was first used for link and server scheduling in [125] and [24] and has since become a promising solution for dealing with stability in various network domains [73, 79, 82]. In a more general case when the arrival rate is either inside or outside of the capacity region, the Lyapunov drift-plus-penalty technique is used to solve joint network stability and utility optimization problems [41, 81, 101]. It is shown that although the max-weight algorithm can achieve throughput optimality, it results in high network delay due to the long queue length [101].

Recently, reinforcement learning has been actively used to tackle network problems such as network traffic and resource control [14, 30, 34, 139]. In [139], a deep Q-learning algorithm based on recurrent neural networks was proposed to learn a scheduling solution for queuing delay optimization by interacting with the environment. In [34], an RL-based scheduling framework was proposed that is capable of

selecting different scheduling rules based on the instantaneous scheduler state to minimize packet delays and packet drop rates for applications with strict quality of service (QoS) requirements. In [30], a new learning-based proactive resource-sharing policy was proposed for next-generation core communication networks. It aims to proactively allocate available forwarding resources on switches to traffic flows to maximize the efficiency of resource utilization with delay satisfaction. The work in [14] proposed a learning-based resource management algorithm that tackles the large queue backlog problem of the max-weight algorithm while achieving throughput optimality. However, most RL works may not be always suitable for addressing the discrete and high-dimensional action spaces in our formulated multi-objective optimization problem with delay guarantees and utility optimality.

3.3 System Model and Problem Formulation

3.3.1 Network Model

We consider the downlink scheduling problem of a single-hop network as shown in Figure 3.1, which consists of one base station (BS) and U ground users. Let $\mathcal{U} = \{1, \cdots, U\}$ denote the set of users that are associated with the BS.

The single-hop network is assumed to operate in discrete time with normalized time slots $t \in \{0, 1, \cdots\}$. All random arriving packets are queued separately for transmission over each downlink. We define the set of packets as $\mathcal{K} = \{1, \cdots, K\}$ which comprises three classes of packets with different delay requirements denoted by $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{L}$, i.e., $\mathcal{K} \triangleq \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}$. The packet arrival rate $\mu_u(t)$ for each user is assumed to be independent and identically distributed over time slots. As shown in Figure 3.1, we consider admission control before injecting each packet into the corresponding buffer. Let $\mathbf{A}(t) \in \{0, 1\}_{U \times 1}$ denote the packet admission vector, where $A_u(t) = 1$ if the packet is injected to queue u at time slot t and $A_u(t) = 0$ otherwise. For convenience, we assume that each packet has the same size and that

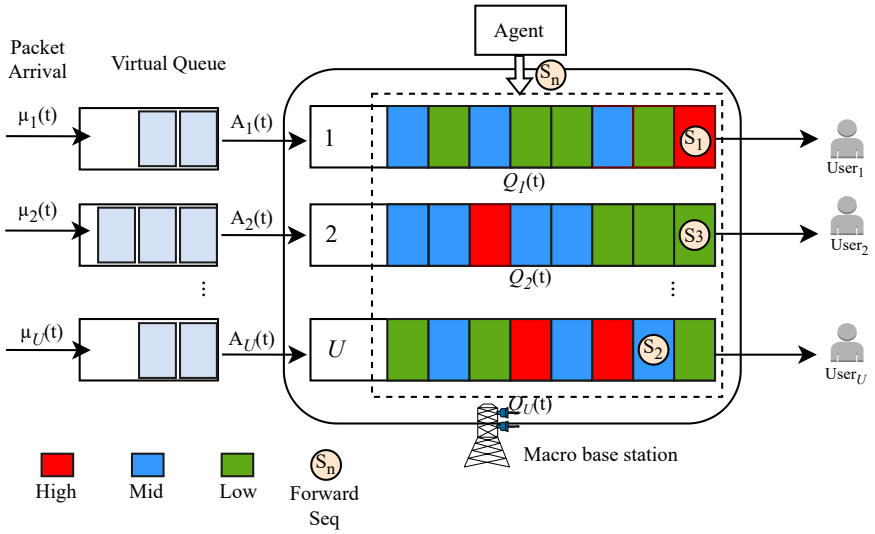


Figure 3.1: Delay-aware selective scheduling.

at most one packet arrives at each downlink at each time slot, so $\sum_{u\in\mathcal{U}} A_u(t) \leq U$.

Note that, unlike most works that consider the first-in-first-out (FIFO) queue management, the packet scheduler of the BS (i.e., agent) in the proposed DASS model selects a suitable packet from all queues to forward by jointly considering link conditions and network-utility maximization[1].

3.3.2 Association and Transmission Model

Let $\mathbf{S}(t) = \{S_1(t), \cdots, S_U(t)\}$ denote the user association vector, and its element $S_u(t) = 1$ if user u associates with the BS at time slot t and $S_u(t) = 0$, otherwise. For a given wireless channel, we assume that the BS serves one user at each time slot, which yields the following constraints

$$\begin{aligned} &S_u(t) \in \{0, 1\}, \ \forall u, t, \\ &\sum\nolimits_{u\in\mathcal{U}} S_u(t) = 1, \ \forall t. \end{aligned} \tag{3.1}$$

For simplicity, we consider the scheduling for a single channel, i.e., at most one

[1]The implementation of DASS mechanism can be achieved by using a set of priority FIFO queues for each user such that packets of the same priority are backlogged in the same buffer and their delay-laxity follows an ascending order.

packet can be transmitted per time slot, which can be easily extended when there are a number of orthogonal channels. Accordingly, we introduce discrete binary transmission variables $\eta_{u,k}(t) \in \{0,1\}$ to indicate the transmission status of packet k of queue u at time t. Then the transmission constraints are given by

$$\sum\nolimits_{k\in\mathcal{K}} \eta_{u,k}(t) \leq 1,\ \forall u \in \mathcal{U}, k \in \mathcal{K}. \tag{3.2}$$

Note that the probability of successful packet transmission over each downlink is affected by the channel condition of the corresponding downlink. We use $\mathbf{C}(t) = \{C_1(t), \cdots, C_U(t)\}$ to denote the channel condition vector, which is assumed to be known by the BS at the beginning of each time slot. For given vectors $\boldsymbol{\eta}(t)$ and $\mathbf{C}(t)$, the probability of successful packet transmission over downlink u is expressed as

$$Pr(\text{downlink to } u \text{ success}|\boldsymbol{\eta}(t), \mathbf{C}(t)) = \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \tag{3.3}$$

where the probability function $\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for $u \in \mathcal{U}$ takes only real values between 0 and 1. Moreover, we introduce an indicator variable $1_u(t)$ to indicate the successful transmission of downlink u:

$$1_u(t) = \begin{cases} 1, & \text{with probability} \quad \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \\ 0, & \text{with probability} \quad 1 - \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)). \end{cases} \tag{3.4}$$

Then the discrete transmission variable $\eta_{u,k}$ in (3.2) can be rewritten as[2]

$$\hat{\eta}_{u,k}(t) = \eta_{u,k}(t) 1_u(t). \tag{3.5}$$

[2]Since the maximum utility is independent of interlink success correlations [96], it suffices to use only the marginal distribution function $\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for each $u \in \mathcal{U}$.

3.3.3 Queuing Model

We define $\mathbf{Q}(t) = \{Q_1(t), \cdots, Q_U(t)\}$ as the set of packets currently backlogged in each queue. Let $x_{uk}(t)$ denote the sojourn time of packet k in queue u at time slot t. In addition, the maximum tolerable queuing delay for the three classes of packets are denoted by Th_H, Th_M, and Th_L, respectively. Packet k will be dropped when its sojourn time in queue u is larger than its maximum tolerable queuing delay. Thus, we introduce a binary delay outage drop variable $D_{u,k}(t)$ to indicate the drop state of each packet, i.e.,

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \mathrm{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \mathrm{Th}_k, \end{cases} \quad \forall k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}. \tag{3.6}$$

To sum up, the queuing dynamics of queue u, including forwarding packet j to the corresponding user while dropping the delay outage packets in the queue, is expressed as

$$\begin{aligned} Q_u(t+1) = \max\{Q_u(t) - S_u(t)\hat{\eta}_{u,j} - \sum_{i \neq j, i \in Q_u(t)} D_{u,i}(t), 0\} \\ + A_u(t), \quad \forall i, j \in Q_u(t). \end{aligned} \tag{3.7}$$

3.3.4 Scheduling and Gain Model

We define the delay laxity [106] as follows:

$$L_{uk}(t) = \mathrm{Th}_k - x_{uk}(t), \forall k \in Q_u(t), \ \forall t, k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}, \tag{3.8}$$

which measures the remaining queue delay budget. To reduce the packet drop rate per queue, we expect to select the packet with the minimum delay-laxity for transmission. Since the BS only transmits at most one packet per time slot, we can obtain different output gains after successful transmission of different classes of packets. We define the gain weights of packets with high, medium and low delay requirements as ω_H, ω_M

and ω_L, respectively, where $\omega_H > \omega_M > \omega_L$. In addition, we consider the discounted potential output gain for all packets backlogged in each queue, which is given as

$$G_u(t) = \rho \sum_{k \in X_u(t)} r_{u,k}(t),\ \forall u \in \mathcal{U}, k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}, \tag{3.9}$$

where $r_{u,k}(t) = \frac{\omega_k}{L_{uk}(t)}$ is the transmission gain of packet k in queue u; $\rho \in [0,1]$ denotes the discount factor; and $X_u(t)$ denotes the set of remaining packets in queue u per time slot after the packet-drop decision is executed. If packet k in queue u is served by the BS in time slot t, the gain received by queue u can be expressed as

$$G_{u,k}(t) = r_{u,k}(t) + \rho(\sum_{i \neq k, i \in X_u(t)} r_{u,i}(t) + \sum_{u' \neq u, u' \in U} \sum_{j \in X_{u'}(t)} r_{u',j}(t)). \tag{3.10}$$

Hence, the average achievable gain of all queues at each time slot can be expressed as

$$G(t) = \frac{1}{U} \sum_{u \in \mathcal{U}} \left[\sum_{k \in X_u(t)} \frac{\omega_k}{L_{uk}(t)} (S_u(t)\hat{\eta}_{u,k} + S_u(t)\rho(1 - \hat{\eta}_{u,k}) + \sum\nolimits_{u=1}^{U} (1 - S_u(t))\rho) \right]. \tag{3.11}$$

3.3.5 Problem Formulation

We define a scheduling problem that considers per-packet delay requirement, network utility, average queuing delay, and admission fairness among users. The gain model introduced in the previous section is defined as the network utility function. Let $F(\bar{\boldsymbol{A}})$ be a concave and non-decreasing function of the U-dimensional vector $\bar{\boldsymbol{A}} = \{\bar{A}_1, \cdots, \bar{A}_U\}$, where $\bar{A}_u$ is used to denote the time-average admission of queue $u \in \mathcal{U}$ (in unit of packets/slot). The following function is useful for addressing network fairness when attributing $\bar{A}_u$ to be non-negative [94]:

$$F(\bar{\boldsymbol{A}}) = \sum_{u=1}^{U} F_u(\bar{A}_u) = \sum_{u=1}^{U} \log(1 + \nu_u \bar{A}_u), \tag{3.12}$$

where ν_u is a positive constant. This example is useful because each component function $\log(1+\nu_u \bar{A}_u)$ has a diminishing return property as $\bar{A}_u$ increases and becomes 0 when $\bar{A}_u = 0$. The average queuing delay is proportional to the average queue length. Therefore, we formulate a multi-objective optimization problem aiming to minimize the average queue length while maximizing the average output gain under the constraints of achieving fairness among users and guaranteeing per-packet delay. Mathematically, this problem is written as

$$\text{P1:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \left\{ -\frac{1}{T}\sum_{t=0}^{T}\sum_{u\in\mathcal{U}} \mathbb{E}[Q_u(t+1)],\ \frac{1}{T}\sum_{t=0}^{T} G(t) \right\} \tag{3.13a}$$

$$\text{s. t} \quad \sum_{u=1}^{U} F_u(\bar{A}_u) \geq \sum_{u=1}^{U} F(\bar{\gamma}_u), \tag{3.13b}$$

$$\bar{A}_u \leq \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}[S_u(t)\hat{\eta}_{u,k}(t)], \forall u \in \mathcal{U} \tag{3.13c}$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U},\ \sum\nolimits_{u\in\mathcal{U}} S_u(t) = 1, \tag{3.13d}$$

$$\eta_{u,k}(t) \in \{0,1\}, \sum_{k\in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{3.13e}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \forall k \in X_u(t). \tag{3.13f}$$

where $\bar{A}_u = \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}[A_u(t)]$ denotes the average number of packets admitted to queue u overall time slots and $\bar{\gamma}_u \in \{0,1\}$ denotes the auxiliary variable that is important for achieving fairness among users with random arrival rate [3]. Constraint (3.13c) guarantees the mean rate stability. In addition, constraint (3.13d) indicates that the BS only serves at most one user at each time slot while constraint (3.13e) is used to limit the number of packets transmitted by the BS in each time slot. Constraint (3.13f) shows the value of $D_{u,k}(t)$ should be either 0 or 1.

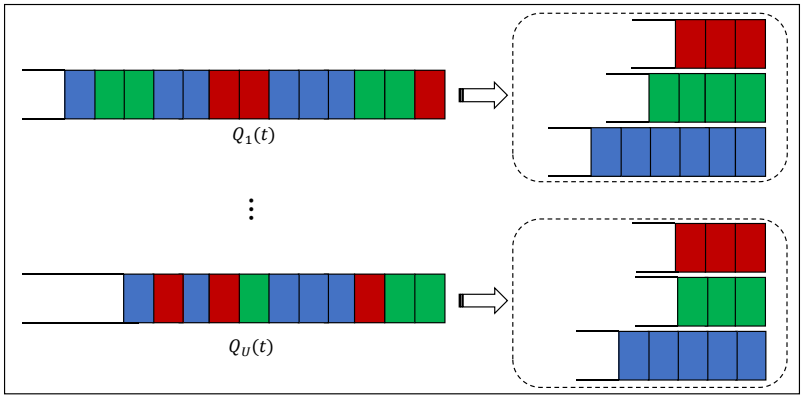


Figure 3.2: Implementation of priority-based single-queue packet selection

3.3.6 Reduced-complexity Scheduling Model

The implementation of the proposed scheduling model may be inefficient in practice, especially for queue maintenance and packet selection. To address this issue, the single-queue model can be replaced by a set of FIFO priority queues as shown in Figure 3.2. The number of priority queues is determined by the delay budget of packets with different delay requirements such that packets in the same priority queue follow the sequence of delay laxity L_{uk} from small to large. Instead, the output gain $r_{u,k}$ of packets in each queue follows a descending order from large to small since the weight w_k of the same queue remains the same. Thus, only the first packet of the same queue needs to be considered for scheduling in each packet-selection process. In addition, only the head of queues should be checked for dropping as well. In our example, three priority queues are designed where $w_{u,H} > w_{u,M} > w_{u,L}$. Furthermore, packets enqueued in the same queue have the same delay budget, i.e., $Th^k_{u,1} = Th^k_{u,2} = \ldots = Th^k_{u,Q_{u,k}}$ for $k \in \{H, M, L\}$, where $Q_{u,k}$ is the queue length of priority queue k in downlink u. The potential sojourn time of packet i in priority queue k of downlink u is denoted by $x^k_{u,i}$ for $i = 1, 2, \cdots, Q_{u,k}$, which follows $x^k_{u,1} > x^k_{u,2} > \cdots > x^k_{u,Q_{u,k}}$, while the output gain of all packets in this queue has $r^k_{u,1} > r^k_{u,2} > \cdots > r^k_{u,Q_{u,k}}$. Note that the number of queues with the same weight can be further extended to meet the time granularity requirements at the cost of memory space. However, the problem formulation remains the same as shown in P1.

3.4 DRL-based Throughput and Delay Optimality

The presented problem P1 is difficult to solve mainly pertaining to the following reasons. Firstly, the downlink selection and packet scheduling variables are binary, and thus (3.13d) and (3.13e) involve integer constraints. Secondly, in problem P1, the first objective minimizing the average queue length and the second objective maximizing the output gain involve conflicts. Although there are several classical optimization algorithms (e.g., dynamic programming and multi-objective genetic local search) that can be used to solve this problem, they have high computational complexity, especially in large-scale scenarios. In addition, due to the curse of uncertainty (i.e., random arrival of packets), it is difficult and impractical to solve using dynamic programming-based algorithms in practice. To this end, we propose a deep reinforcement learning (DRL)-based method to solve this multi-objective optimization problem, which is used to learn policies by interacting with the environment without any prior knowledge to maximize the cumulative rewards from experiences. Specifically, we decompose problem P1 into a set of scalar optimization subproblems by using the weighted sum approach [27]. We then model each subproblem as a partially observable Markov decision process (POMDP) and explore an efficient double deep Q network (DDQN)-based algorithm to solve it. In particular, we collaboratively optimize the network parameters of all subproblems by applying the neighborhood parameter transfer method [37] and the proposed DDQN-based training algorithm.

3.4.1 DRL for Multi-objective Optimization

In this subsection, we decompose the multi-objective optimization problem into a set of scalar subproblems. There are many scalarizing methods that can be used for decomposition, such as the weighted sum method and the penalty-based boundary intersection method [66]. For simplicity, we use the weighted sum method in which a set of uniformly distributed vectors $\lambda^1, \cdots, \lambda^N$ is given, e.g., $(1,0), (0.99, 0.01), \cdots, (0,1)$ for a bi-objective problem. Note that $\lambda^n = \{\lambda_{n1}, \cdots, \lambda_{nL}\}$, where L is the number

of objectives. Thus the original multi-objective problem is transformed into N scalar subproblems and each subproblem is solved collaboratively with other subproblems through the neighborhood-based parameter transfer strategy. Note that solving each scalar subproblem usually results in a Pareto optimal solution and the desired Pareto Front can be obtained when all the scalar optimization subproblems are solved in sequence. Particularly, the objective function of the n-th subproblem is expressed as

$$\max \lambda^n \times f = \sum_{l=1}^{L} \lambda_{nl} \times f_l. \tag{3.14}$$

To solve these subproblems by DRL, we model each of them as a neural network. Then the N scalar subproblems are solved collaboratively according to the neighborhood-based parameter transfer strategy and the proposed DDQN-based algorithm. From (3.14), two neighboring subproblems may have very close optimal solutions since their weight vectors are adjacent [126]. In this case, each subproblem can be solved faster by leveraging the knowledge of its neighboring subproblems. In specific, the parameters of the neural network model of the $(n-1)$-th subproblem is described as $[\omega_{\lambda^{n-1}}, b_{\lambda^{n-1}}]$, where $[\omega^*, b^*]$ denotes the optimal parameters of the neural network model and $[\omega, b]$ denotes the parameters that have not been optimized. Once the $(n-1)$-th subproblem has been solved, i.e., its network parameters obtained by the proposed algorithm are close to optimal. The best network parameters $[\omega_{\lambda^{n-1}}, b_{\lambda^{n-1}}]$ obtained for the $(n-1)$-th subproblem are set as the starting point of the network training for the n-th subproblem. Accordingly, the network parameters are transferred sequentially from the previous subproblem to the next subproblem as shown in Figure 3.3, which can save a considerable amount of time compared with training all subproblems.

The DRL framework for solving our proposed optimization problem P1 is summarized in Algorithm 4, which consists of the multi-objective decomposition and the neighborhood-based parameter transfer strategy. In this algorithm, each subproblem is modeled as a POMDP and solved by the proposed DDQN algorithm and all the

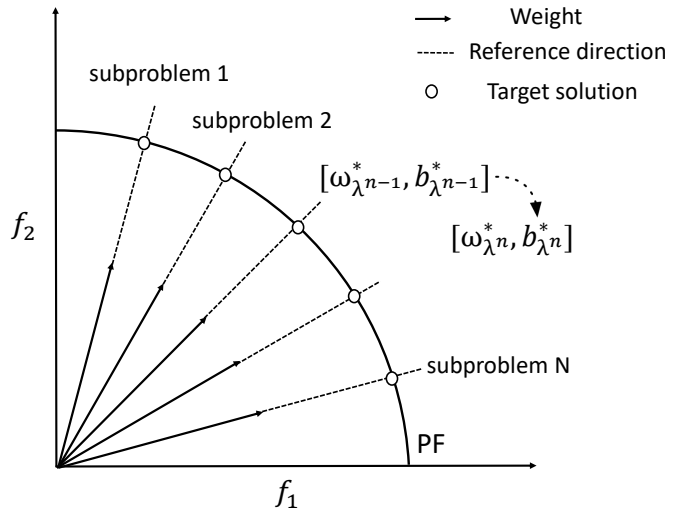


Figure 3.3: Illustration of the decomposition and parameter-transfer strategy.

subproblems can be solved sequentially by transferring the network weights. Hence, we can obtain an approximate Pareto solution by solving each subproblem using Algorithm 4. Finally, the desired Pareto Front is obtained when all the subproblems are solved sequentially.

The proposed DRL framework has two main advantages: i) The first is its simplicity and modularity for use, i.e., any of the recently proposed novel DRL-based solvers (including the improved DQN algorithm) can be integrated into the proposed DRL framework to solve multi-objective optimization problems; ii) Once the trained model is available, we can obtain the desired Pareto Front directly by a simple forward propagation of the model.

The DRL framework acts as an outer loop. The next issue is how to model and solve those decomposed scalar subproblems. In the next subsection, we first formulate each subproblem as a POMDP and then resort to a value iterative-based reinforcement learning algorithm, named Q-learning, to solve it.

3.4.2 Preliminaries of POMDP

In our learning framework, we consider the controller on the BS as an agent to learn how to perform data admission control, downlink service scheduling, and packet selection in the system. In addition, since the admission and delivery of each packet may be affected by the current network environment and the actions of the BS, the learning task of the BS agent can satisfy the Markov property. Thus, we model each scalar optimization subproblem as a POMDP, which is described by the following tuple:

$$\Omega = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}, \tag{3.15}$$

where $\mathcal{S}$ denotes the set of states describing the environment; $\mathcal{O}$ denotes the observation space; $\mathcal{A}$ denotes the action space; $\mathcal{R}$ denotes the reward function that maps the network state and the joint actions of the agent to rewards; $\mathcal{P}$ denotes the state transition function with $P_{s_t,s_{t+1}}(a_t)$ being the probability that the current state s_t transfers to the next state s_{t+1} when action a_t is performed; and $\gamma \in [0, 1]$ denotes the discount factor. At time slot t, the agent observes a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$ according to a certain policy $\pi : \mathcal{S} \to \mathcal{A}$, which receives a reward $r_t = r(s_t, a_t)$ and produces a new state s_{t+1} with the transition probability $P(s_{t+1}|s_t, a_t)$.

The detailed definitions of POMDP for the n-th subproblem with weight $\lambda_n = (\lambda_{n1}, \lambda_{n2})$ are given below.

State and Observation Space

At each time slot t, the BS-agent observes the state information of the environment so as to determine the corresponding policy. The state space at time slot t is denoted by s_t and it contains four elements: the successful transmission probability $1_u(t)$, queue length $Q_u(t)$, packet sojourn time $x_{u,k}(t)$ and delay budget laxity $L_{uk}(t)$, which can be expressed as

$$s_t = \{1_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}, \forall u \in \mathcal{U}, k \in Q_u(t). \tag{3.16}$$

Algorithm 4 DRL Framework for Multi-objective Optimization Problems.

Input: The model of the subproblem $M = [\omega, b]$ and weight vectors $\lambda^1, \ldots, \lambda^N$
Process:
Random initialize $[\omega_{\lambda^1}, b_{\lambda^1}]$;
for $n = 1, \ldots, N$ **do**
  if $n == 1$ **then**
    Solve the first subproblem using DDQN algorithm and obtain the network parameters $[\omega^*_{\lambda^1}, b^*_{\lambda^1}]$
  else
    $[\omega^*_{\lambda^n}, b^*_{\lambda^n}] \leftarrow [\omega^*_{\lambda^{n-1}}, b^*_{\lambda^{n-1}}]$
    Solve the n-th subproblem using DDQN algorithm and obtain the network parameters $[\omega^*_{\lambda^n}, b^*_{\lambda^n}]$
end for
Output: Pareto Front can be directly computed by $[\omega^*, b^*]$

Thus the state space is expressed as $\mathcal{S} = \{s_t | t = 1, \cdots, T\}$. For the state space, the probability of successful transmission for each user depends on the channel conditions that can only be observed locally and not known by other association pairs. Therefore, according to (3.4), the observation space at time slot t can be summarized as

$$o_t = \{C_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}, \forall u \in \mathcal{U}, k \in Q_u(t). \tag{3.17}$$

Accordingly, the observation space of the BS-agent is given by $\mathcal{O} = \{o_t | t = 1, \cdots, T\}$.

Action Space

At each slot, the BS-agent determines whether and what types of packets are admitted in each queue. Then the BS chooses which user is served and which packet is transmitted in the queue corresponding to the associated user. In addition, if the sojourn time of the packet exceeds its maximum tolerable queuing delay, it will be dropped. Thus, the action of the BS-agent at time slot t is expressed as

$$a_t = \{A_{u,k}(t), S_u(t), \eta_{u,k}(t), D_{u,k}(t)\}, \forall u \in \mathcal{U}, k \in \mathcal{K}. \tag{3.18}$$

Thus the action space is expressed as $\mathcal{A} = \{a_t | t = 1, \cdots, T\}$.

Reward Design

In a DRL-based framework, the learning process is driven by the generated reward and the agent makes its policy decision by interacting with the environment in terms of maximizing the designed reward. Thus, the reward design is crucial for problems with multiple objectives, while the performance of the system largely depends on the reward. It is obvious that network reward is generally related to the objective function. According to the presented problem P1, our objective is twofold: minimizing the average queue length and maximizing the average output gain. As a result, the immediate network reward is defined as

$$r(s_t, a_t) = \lambda_{n2}\frac{1}{T}\sum_{t=0}^{T} G(t) - \lambda_{n1}\frac{1}{T}\sum_{t=0}^{T}\sum_{u\in\mathcal{U}} \mathbb{E}[Q_u(t+1)]. \tag{3.19}$$

As shown in Figure 3.4, the learning system establishes the relationship between the optimal criterion and the optimal policy by introducing a value function consisting of a state-value function and an action-value function. Specifically, the state-value function $V_\pi(s)$ is defined by the discounted cumulative reward $R_t = \sum_{t=0}^{T} \gamma r_t$ of the agent in state s, which is used to measure the quality of an available state-action pair. Given a policy π, we define the state-value function as

$$V_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s] = \mathbb{E}_\pi[(\sum_{t=0}^{T} \gamma r_t)|s_t = s], \tag{3.20}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expectation under the policy π. Based on the Bellman equation [52], $V_\pi(s)$ is converted as follows

$$V_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)(r(s,a) + \gamma \sum_{s'} P_{s,s'}(a)V_\pi(s')), \tag{3.21}$$

where $\pi(a|s)$ is the action distribution under state s and s' is the state at the next time slot. Similarly, the Q-value function is defined as the expected sum of discounted rewards obtained by performing action a at state s and following policy π in the next

state, which is given by

$$Q_{\pi}(s,a) = r(s,a) + \gamma \sum_{s'} P_{s,s'}(a) \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_{\pi}(s', a'). \tag{3.22}$$

There exists an optimal state-value function when the optimal policy is used among all of those possible state-value functions. We use $V^*(s) = \max_{\pi} V_{\pi}(s)$ to denote the optimal state-value function. Moreover, the optimal state-value function $V^*(s)$ in state s can be estimated by the Q-value function $Q(s,a)$. Thus we have

$$V^*(s) = \max_{a} Q(s,a). \tag{3.23}$$

The agent continuously improves its policy with the accumulation of experience to search an optimal policy that yields a maximum value of $Q(s,a)$ for all available states and actions. Overall, the optimal Q-value function is easily obtained when the optimal policy $\pi^*(s) = \max_{\pi} Q_{\pi}(s,a)$ that maps the set of states and actions is satisfied. The Bellman optimality equation is used to express the optimal Q-value function, which can be mathematically written as

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P_{s,s'}(a) \max_{a'} Q^*(s', a'), \tag{3.24}$$

In addition, the optimal Q-value function can be estimated by iteratively updating the following expression at each time slot based on the recursive method.

$$Q_{\text{new}}(s,a) = (1-\alpha) Q(s,a) + \alpha (r(s,a) + \gamma \max_{a'} Q(s', a')), \tag{3.25}$$

where $\alpha \in (0,1)$ denotes the learning rate.

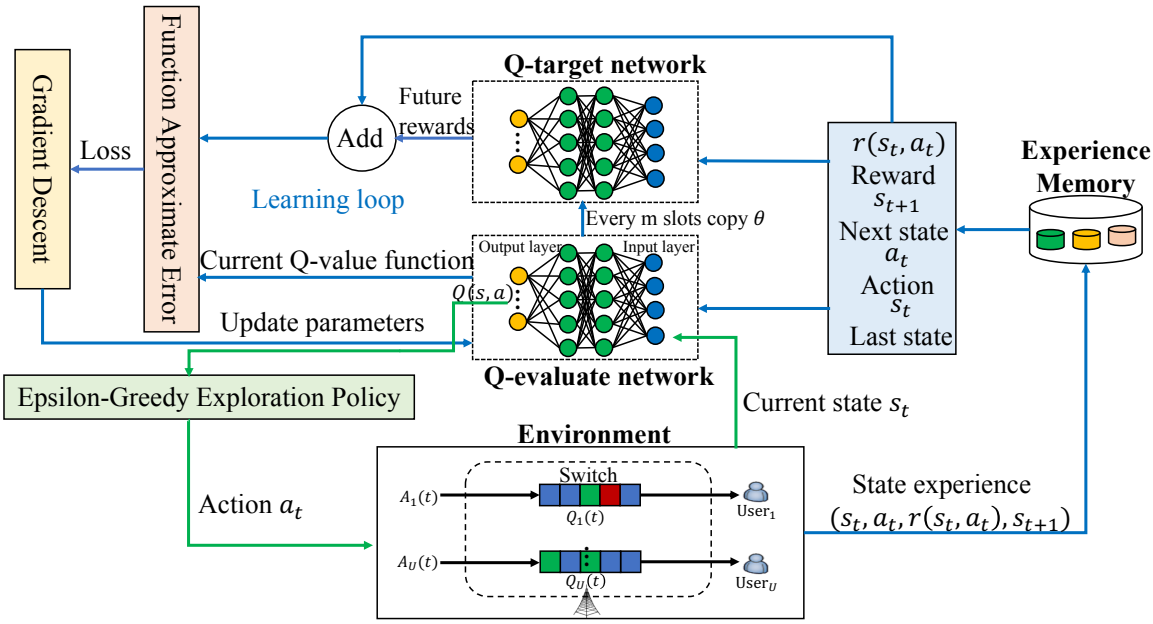


Figure 3.4: Illustration of DDQN framework for resource scheduling in networks.

3.4.3 Optimize with Lyapunov Function

We consider two cases where the traffic is inside or outside of the capacity region, i.e., underload or overload. For the first case, the admission rate is equal to the arrival rate and then the network utility is maximized with an optimal scheduling policy that transmits as many packets as possible under the network stability constraint [102]. Moreover, all queues in the network can achieve stability simultaneously by maximizing the weight difference between the output gain and the average queue length, which is demonstrated in Proposition 1. Inspired by these facts, if the optimal objective (3.13a) is achieved, both constraints (3.13b) and (3.13c) are satisfied simultaneously. Thus, the original problem P1 is converted into the following form by introducing a negative Lyapunov drift term $-\nu_1(Q_u(t+1)^2 - Q_u(t)^2)$ into the objective function

$$\text{P2:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \Big\{ \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) + \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[-Q_u(t+1) - \nu_1 (Q_u^2(t+1) - Q_u^2(t))] \Big\} \tag{3.26a}$$

$$\text{s. t} \quad \nu_1 > 0, \tag{3.26b}$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U}, \ \sum\nolimits_{u\in\mathcal{U}} S_u(t) = 1, \tag{3.26c}$$

$$\eta_{u,k}(t) \in \{0,1\}, \sum_{k\in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{3.26d}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \mathrm{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \mathrm{Th}_k, \end{cases} \forall k \in X_u(t). \tag{3.26e}$$

Although the objective function is changed, we can still obtain an optimal solution that maximizes the average output gain and minimizes the average queue length while making all queues in the network stable. It is clear that the Lyapunov drift term is the increment of queue backlog from time slot t to the next time slot $t+1$. As expected, each queue can be pushed to a low congestion state by minimizing it, which guarantees network stability [125]. Moreover, all queues will not diverge if they are stable, which indicates that the drift term converges to 0 as t tends to infinity and the objective function leaves only the average output gain and the average queue length.

Proposition 1 *There exists an optimal policy for problem P2 that makes all queues in the network stable under the inner capacity region while maximizing the average output gain and minimizing the average queue length*[3].

Proposition 1 indicates that the optimal solution of problem P1 is the same as that of problem P2.

Next, we consider a more general case where the traffic is outside of the capacity region. Since all users in our model share limited resources, it is essential to solve the network utility maximization problem in order to fairly allocate network resources while stabilizing the network. For this overload case, the admission rate will no longer be equal to the arrival rate. According to the Lagrangian method [8], the original

[3]See the Appendix for proof.

problem P1 is transformed as follows

$$\text{P3:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \left\{ \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)] + \nu_2 \sum_{u=1}^{U} F_u(\bar{A}_u) \right\} \tag{3.27a}$$

$$\text{s. t} \quad \nu_2 > 0, \tag{3.27b}$$

$$\bar{A}_u \leq \frac{1}{T} \sum\nolimits_{t=0}^{T} \mathbb{E}[S_u(t) \hat{\eta}_{u,k}(t)], \forall u \in \mathcal{U}, \tag{3.27c}$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U}, \ \sum\nolimits_{u \in \mathcal{U}} S_u(t) = 1, \tag{3.27d}$$

$$\eta_{u,k}(t) \in \{0,1\}, \sum_{k \in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{3.27e}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \forall k \in X_u(t). \tag{3.27f}$$

According to the two transformed problems P2 and P3, the reward function (3.19) is rewritten as

$$r(s_t, a_t) = \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) + \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[-Q_u(t+1) - \nu_1 (Q_u^2(t+1) - Q_u^2(t))] \tag{3.28}$$

for the arrival rate inside the capacity region and

$$r(s_t, a_t) = \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)] + \nu_2 \sum_{u=1}^{U} F_u(\bar{A}_u) \tag{3.29}$$

for the arrival rate lying outside the capacity region.

3.5 Algorithm Design

Although Q-learning has emerged as a prospective learning algorithm based on value, it relies heavily on a set of records in the sample when searching the optimal policy.

Therefore, it is susceptible to the training variance and even the convergence of the Q-value function. The work [10] showed that Q-learning algorithms are very flexible for low-dimensional reinforcement learning problems. However, in our considered time-varying model, the complex and large state-action space can easily trap the table of Q-value functions in the curse of dimensionality, which will waste a lot of time and resources and even exceed the memory size for maintaining this table.

In order to address the above problems, many scholars have focused on the deep Q-network (DQN) algorithm based on the combination of neural networks and Q-learning, which mainly takes the state and action as the input of the neural network to approximate the Q-value function instead of maintaining the Q-table [58]. To be specific, the Q neural network receives an input state s and outputs the estimated Q-value functions for all optional actions, i.e., $Q(s,a;\boldsymbol{\theta}) \approx Q(s,a)$, $a \in A$, where $\boldsymbol{\theta}$ represents the vector of the weights of a Q neural network. In this algorithm, the agent aims to continuously learn an optimal policy for optimizing the Q-value function by minimizing the loss function $\text{Loss}(\boldsymbol{\theta})$, which is expressed as

$$\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(y(t) - Q(s,a;\boldsymbol{\theta}))^2], \tag{3.30}$$

where $y(t)$ denotes the target Q-value and can be mathematically written as

$$y(t) = r(s,a) + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}). \tag{3.31}$$

According to (3.30), the Q neural network is trained by iteratively updating its weights $\boldsymbol{\theta}$ to optimize the approximation of the Q-value function. The parameter update equation of the Q neural network can be written as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \mathbb{E}[(y(t) - Q(s,a;\boldsymbol{\theta}))\nabla Q(s,a;\boldsymbol{\theta})]. \tag{3.32}$$

It is worth noting that the DQN-based framework has an experience replay pool and is

usually used to store the experiences collected by the agent denoted by (s_t, a_t, r_t, s_{t+1}), from which a group of experiences can be randomly selected and used to train the neural network. Moreover, this framework employs a dual network approach to further improve the learning stability and algorithm efficiency, which means that there is also a target network with weights $\boldsymbol{\theta}^{'}$ in the training process. Accordingly, the target Q-value $y(t)$ in (3.31) can be described in a new way as follows

$$y(t) = r(s, a) + \gamma \max_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}^{'}). \tag{3.33}$$

It is easy to see from the Q-value function that DQN uses a single max mathematical estimator to select and evaluate an action. However, this manner tends to make agents sometimes confused about the selection and evaluation of actions, which leads to estimated Q-values that may be higher than the true values. Note that these overestimation problems are caused by the positive bias of the max operator used in DQN to update its Q-value function.

3.5.1 DDQN-based Solution

To address this overestimation problem, we explore a double deep Q network (DDQN)-based method, which decouples the action selection and Q-value calculation into two separated max function estimators to avoid overestimation [56]. Evidently, the maximum Q-value function is computed by the state of the next time slot and all possible actions in the current neural network, DDQN first finds the optimal action under that Q-value function and then uses this action to predict the target Q-value from the target network. Due to the mutual constraints between the dual estimators, they can eliminate the maximum deviation.

Similar to DQN, DDQN also has two neural networks namely online network $Q(s, a; \boldsymbol{\theta})$ and target network $(Q(s, a; \boldsymbol{\theta}^{'})$. The weight vector of the target network $\boldsymbol{\theta}^{'}$ is copied from the online network in several previous iterations. Inspired by this, the

target Q-value for DDQN is replaced as follows

$$y(t) = r(s,a) + \gamma Q(s^{'}, \arg\max_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}); \boldsymbol{\theta}^{'}). \tag{3.34}$$

Note that the selection and evaluation of an action follows an **arg max** operator, which is defined by a set of weights $\boldsymbol{\theta}$. This means that we estimate the value of the greedy policy based on the current Q-value defined by $\boldsymbol{\theta}$. Then another set of weights $\boldsymbol{\theta}^{'}$ is used to evaluate the value of this policy. The agent selects an action based on the ε-greedy policy to balance its exploitation and exploration, where ε is a diminishing value for exploration. Particularly, the agent selects an action that can maximize the Q-value with probability $(1-\varepsilon)$, while randomly choosing other actions with probability ε, which can be described as

$$a(t) = \begin{cases} \text{random action}, & \text{probability } \varepsilon, \\ \arg\max_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}), & \text{probability } 1-\varepsilon. \end{cases} \tag{3.35}$$

The loss function defined in (3.30) is written as

$$\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(r(s,a) + \gamma Q(s^{'}, \arg\max_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}); \boldsymbol{\theta}^{'}) - Q(s,a;\boldsymbol{\theta}))^2]. \tag{3.36}$$

In order to reduce the loss value, the gradient descent method is used to update the weights of the target network. The update of $\boldsymbol{\theta}$ can be expressed as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha\mathbb{E}[(r(s,a) + \gamma Q(s^{'}, \arg\max_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}); \boldsymbol{\theta}^{'}) - Q(s,a;\boldsymbol{\theta}))\nabla Q(s,a;\boldsymbol{\theta})]. \tag{3.37}$$

3.5.2 Implementation of DDQN-based Algorithm

According to the above analysis, we summarize the detailed procedure of our proposed DDQN-based algorithm for solving problems P2 and P3 as Algorithm 5. An illustration of the training process is shown in Figure 3.5. Before training, we ini-

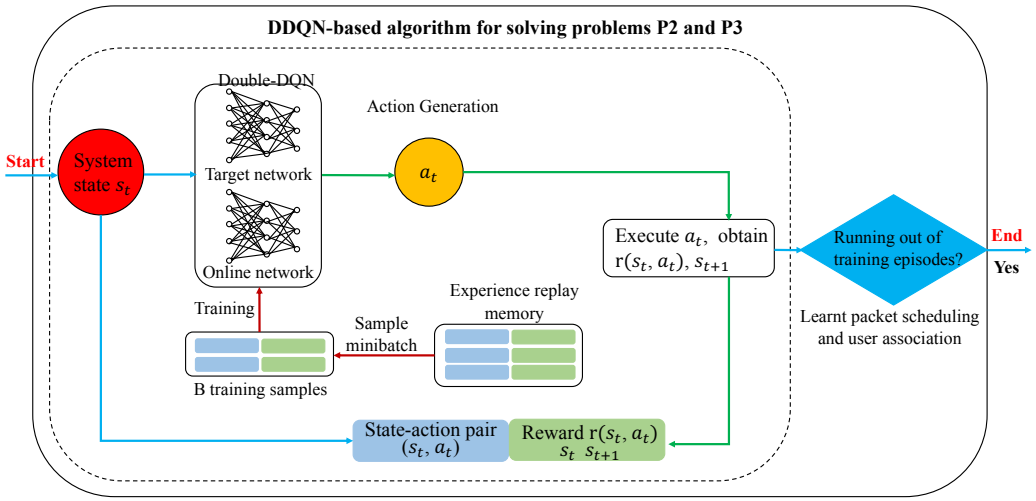


Figure 3.5: Training process of DDQN-based algorithm for solving P1 and P2.

tialize the experience replay memory and the weights of the online network and the target network. In particular, the weight of the target network is initialized to be the same as that of the online network, i.e., $\boldsymbol{\theta}' = \boldsymbol{\theta}$. As shown in Figure 3.5, we develop two neural networks to illustrate the correlation between each state-action pair (s, a) and its value function $Q(s, a)$. Hence, it is crucial to preprocess the admission control and output selection and user association of the downlink network for a sufficiently long time with a random policy.

Algorithm 5 DDQN-based Training Algorithm

Input: Action set $\mathcal{A}$; number of episodes E; learning rate α; network update period F; discount factor γ; minibatch size B; random selection probability ε.
Output: Scheduling policy π^* and maximum reward R
Initialize: Relay memory M with the capacity of C; online Q-network $Q(s, a; \boldsymbol{\theta})$ with weight $\boldsymbol{\theta}$; target Q-network $Q(s, a; \boldsymbol{\theta}^{'})$ with $\boldsymbol{\theta}^{'} = \boldsymbol{\theta}$
for each episode **do**
  Initialize the environment and obtain an initial state s_1
  for each iteration of an episode **do**
    Observe o_t and take an action a_t from $\mathcal{A}$ with a random probability $\hbar$ based on the ε-greedy policy
    if $\hbar < \varepsilon$ **then**
      Randomly choose an action a_t;
    else
      Choose an action $a_t = \arg\max_{a' \in \mathcal{A}} Q(s_t, a^{'}; \boldsymbol{\theta})$;
    end if
    Execute action a_t and receive a reward $r(s, a)$
    Get the next state s_{t+1} for the current action and state
    Update the input to Q_{t+1} based on the observed state
    Store the current experience tuple $\{s_t, a_t, o_t, r_t, s_{t+1}, o_{t+1}\}$ in the experience relay memory
    Randomly choose a minibatch of experiences $\{s_t, a_t, o_t, r_t, s_{t+1}, o_{t+1}\}$ from the experience relay memory
    Calculate the target Q-value by the learning step with optimizer
    if an episode terminates at iteration $t + 1$ **then**
      $y(t) = r(s, a)$;
    else
      $y(t) = r(s, a) + \gamma Q(s^{'}, a^{'}; \boldsymbol{\theta}); \boldsymbol{\theta}^{'})$;
      $a^{'} = \arg\max_{a' \in \mathcal{A}} Q(s_t, a^{'}; \boldsymbol{\theta})$;
    end if
    Update the weight of the online network by minimizing the loss function as
    $\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(y(t) - Q(s, a; \boldsymbol{\theta}))^2]$
    Perform a gradient descent step on $\text{Loss}(\boldsymbol{\theta})$ relative to the weight of the online network $\boldsymbol{\theta}$
    Update the weight of the target network based on the weight of the online network $\boldsymbol{\theta}$
    Rest the target network $Q^{'} \leftarrow Q$ after F iterations
  end for
end for

On this basis, the estimation of $Q(s,a)$ can be obtained by means of some state transition information, which is then stored in the experience replay memory. Moreover, we train the neural network with the input state-action pair (s,a) and the outcome $Q(s,a)$. After that, the action selection and Q-value can be achieved. More specifically, in each decision episode, the network environment is initialized and the agent observes the initial state space $\{C_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}$ from the environment simulator. Then, we adopt the ε-greedy policy with the greedy factor $\varepsilon \in [0,1]$ to select the execution action a. For a random probability $\hbar < \varepsilon$, the online network randomly selects an action from the action space $\mathcal{A}$. Otherwise, the action with the largest Q-value function derived from the online network with the input of the state-action pair (s,a) is selected. After selecting an action a, the environment simulator provides the corresponding reward $r(s,a)$ to the agent and the state is transferred from s_t to the next state s_{t+1}.

To improve training stability, we use the experience replay memory to store the experience tuple $\{s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}\}$ and randomly select a minibatch of B experiences to train the weights of the online network and the target network at each iteration. During the learning process, the online network and the target network update their model weights $Q(s^{'}, a^{'}; \boldsymbol{\theta})$ and $Q(s^{'}, a^{'}; \boldsymbol{\theta}^{'})$, respectively. Given the current reward $r(s,a)$ and the discount factor γ, the target network yields a target value

$$y(t) = \begin{cases} r(s,a), \\ r(s,a) + \gamma Q(s^{'}, \arg\max\limits_{a^{'}} Q(s^{'}, a^{'}; \boldsymbol{\theta}); \boldsymbol{\theta}); \boldsymbol{\theta}^{'}), \end{cases}$$

which is then compared with the estimated value of the online network to obtain the loss value $\mathrm{Loss}(\boldsymbol{\theta})$. Moreover, the weight of the online network is updated by using a gradient descent step of $[\partial \mathrm{Loss}(\boldsymbol{\theta})/\partial \boldsymbol{\theta}]$. The weight of the target network does not need to be updated iteratively and it can copy $\boldsymbol{\theta}$ from the online network after a certain number of iterations. Note that the value of the loss function gradually decreases by constantly updating the weight of the online network. When the loss

value reaches the global minimum, our proposed algorithm outputs the corresponding optimal policy. Figure 3.5 describes the detailed procedure of the proposed DDQN-based algorithm

3.6 SIMULATION RESULTS

In this section, we present simulation results to demonstrate the effectiveness of our proposed DDQN-based algorithm. We first introduce our simulation setup and network architecture. We then compare the proposed algorithm with several other algorithms and analyze the simulation results under different scheduling policies.

3.6.1 Simulation Setup

We consider a single-hop downlink system in which $U = 4$ ground users are associated with a single BS. In this system, the packet scheduling matrix is selected every time slot within the queue corresponding to each user $u \in \mathcal{U}$, so that at most one packet is served per input and per output at each time slot[4]. The arrival process for each queue follows a Bernoulli distribution, which is independently and identically distributed over time slots with the arrival rate r_{uk}. A total of $K = 2000$ packets consisting of various delay requirements, i.e., $H = 400$, $M = 600$ and $L = 1000$, arrives at the BS over a period of time. The maximum tolerable queuing delays of the three classes of packets are set as $\mathrm{Th}_H = 10$, $\mathrm{Th}_M = 20$ and $\mathrm{Th}_L = 30$, respectively, with the unit of time slots. In addition, the output gain weights of the three classes of packets are set as $\omega_H = 0.5$, $\omega_M = 0.3$ and $\omega_L = 0.2$, respectively. The discount factor of the potential output gain is set as $\rho = 0.3$.

We conduct the experimental simulations using a server with an NVIDIA GTX 2080 Ti GPU. The software platform of the experiment is Python 3.6 with PyTorch [108]. Our developed DDQN-based algorithm consists of the online-network and

[4]The proposed solution can be easily extended to the system to schedule packets for multiple orthogonal channels

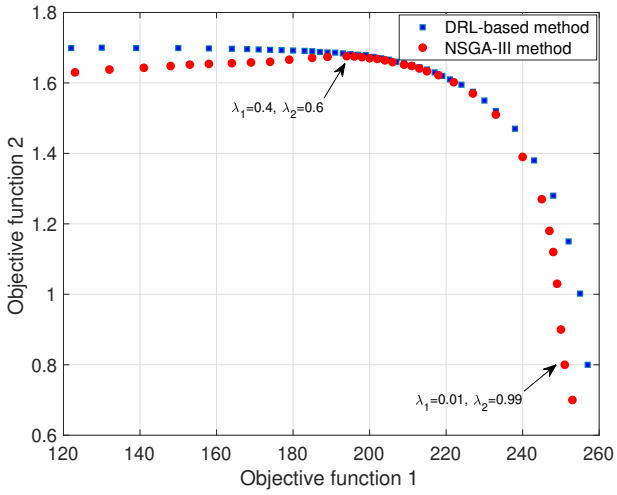


Figure 3.6: Pareto-Front achieved by the proposed DRL-based framework.

target-network, each of which has one input layer, two hidden layers and one output layer. Moreover, each hidden layer is assumed to have the same number of neurons and is defined as $e = 64$. We use the rectified linear unit (ReLU) function $f_{\text{ReLU}}(x) = \max\ \{0, 1\}$ to describe the activation function in each hidden layer. The Adam optimizer is used to update the weights of the online network and the target network. Besides, the target network is updated by the online network every 100 training iterations. The learning rates of both neural networks are set as $\alpha = 0.0001$ to ensure that the training process does not miss all possible local solutions. The discount factor is set as $\gamma = 0.999$. During the training process, the ε-greedy policy is used where the value of ε is set as 0.9 at the beginning and then gradually decreases to 0.1. The training process of our proposed DDQN-based algorithm has $E = 2000$ episodes. The capacity of the experience replay memory is set as $C = 2000$ and the size of each minibatch sampled from this experience replay memory is set as $B = 64$.

3.6.2 Result Analysis

Based on the proposed DRL framework, the multi-objective problem of scheduling and association can be solved by using the neighborhood-based parameter transfer strategy. In Figure 3.6, we illustrate the Pareto front obtained by the proposed DRL-

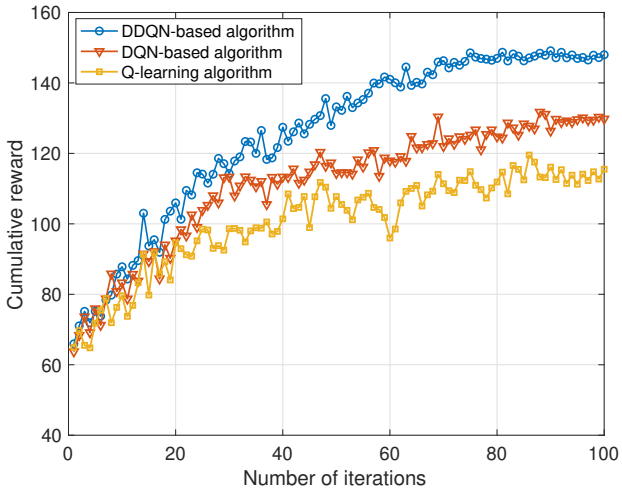
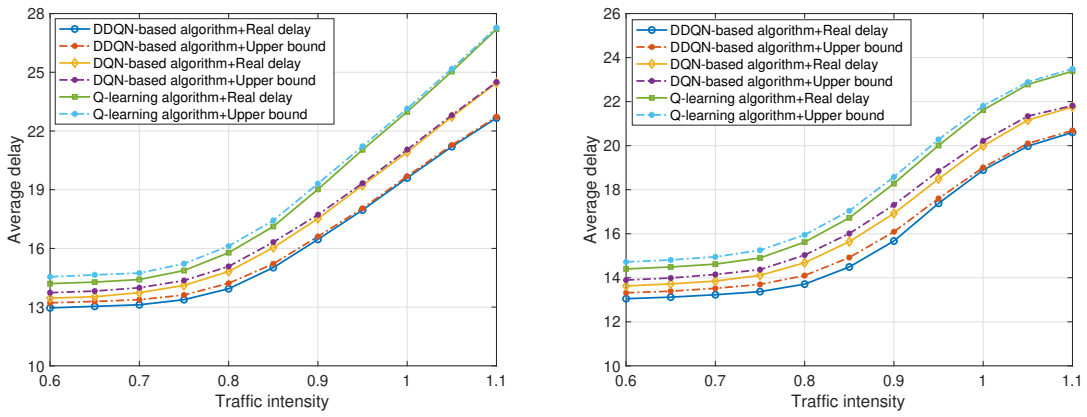


Figure 3.7: Convergence of different algorithms for the overload case.

based method and the NSGA-III method [38]. It can be seen from Figure 3.6 that the performance of the DRL-based method is close to that of the NSGA-III method when the weight values λ_1 and λ_2 are close each other (e.g., $\lambda_1 = 0.4$ and $\lambda_2 = 0.6$). When the difference between λ_1 and λ_2 is significant (e.g., $\lambda_1 = 0.01$ and $\lambda_2 = 0.99$), the DRL-based method tends to spare most of its effort in training only one of the objective functions. The obtained policy will then perform relatively poor on the other objective function. Whereas the DRL-based method shows its advantage over the NSGA-III method in terms of computing time since it only needs a very simple forward progression after the training process is completed. This advantage becomes more prominent when the complexity of the multi-objective problem increases, e.g., the objective contains multiple integer decision variables.

To demonstrate the convergence of the proposed algorithm and the other two algorithms, Figure 3.7 plots the learning curves of these three algorithms in the case where the arrival rate is outside of the capacity region. With the increase of the number of iterations, the cumulative rewards obtained by the three algorithms have an obvious tendency to increase and converge. It is clear that the proposed DDQN-based algorithm always outperforms the other two algorithms in terms of cumulative reward. The reason is that the proposed algorithm can prevent the overestimation



(a) Inner capacity case. (b) Overload case.

Figure 3.8: Two cases of delay versus traffic intensity.

of the action-value function by decoupling the Q-target. Besides, the convergence speed of the proposed DDQN-based algorithm and the DQN-based algorithm is substantially higher than that of the Q-learning algorithm. This is because both the proposed DDQN-based algorithm and the DQN-based algorithm use the neural network $Q(s, a; \boldsymbol{\theta})$ to approximate the target Q-value $Q^*(s, a)$, which can greatly reduce the time of searching the maximum value.

Delay performance

Figure 3.8 (a) shows the delay performance of different algorithms versus traffic intensity ϱ[5] when the arrival rate is within the capacity region. The solid and dash lines indicate the average delay calculated based on the observed delay (real delay) and the maximum tolerable delay per packet (upper bound), respectively. The average delay achieved by our proposed algorithm and the other two algorithms increases with the increase of traffic intensity ϱ and eventually diverges when $\varrho > 1$, i.e., outside the capacity region. In addition, we observe that our proposed DDQN-based algorithm achieves 9% and 20% delay reduction compared with the DQN-based algorithm and the Q-learning algorithm, respectively.

[5]The traffic intensity ϱ is defined as the ratio of the arrival rate to the capacity region boundary [14] to measure the average resource occupancy.

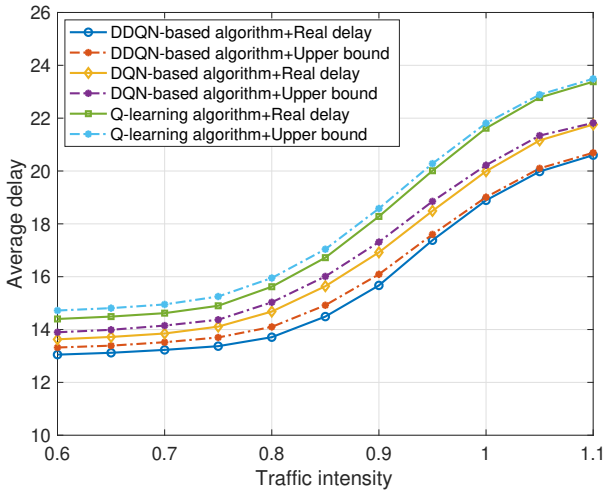


Figure 3.9: Delay versus traffic intensity for different design schemes.

Figure 3.8 (b) shows the delay performance versus traffic intensity ϱ for the general case. As can be seen from Figure 3.8 (b), for the three algorithms, the average delay increases with ϱ and eventually becomes saturated when ϱ is sufficiently large. In addition, our proposed DDQN-based algorithm shows up to 6% and 13% delay reduction compared to the DQN-based algorithm and the Q-learning algorithm, respectively. From Figure 3.8, we observe that our proposed DDQN-based algorithm outperforms the other two algorithms in terms of average delay, which demonstrates the effectiveness of our proposed algorithm. It is obvious that the real average delay gradually approaches the upper bound as ϱ becomes large in both cases. The reason is that as ϱ grows, the queue length of each user increases leading to longer queue delay, and thus packets reach their maximum tolerable queuing time. On the other hand, it shows that our proposed model guarantees per-packet delay.

In order to demonstrate the superiority of the proposed joint-scheduling model, we consider the following three schemes: (i) scheme 1 jointly optimizes user association and packet selection but without admission control [14]; (ii) scheme 2 jointly optimizes admission control and user association but with FIFO queue management [95]; and (iii) scheme 3 adopts random packet admission and user association as well as FIFO queue management. Note that all schemes are solved by the proposed DDQN-based

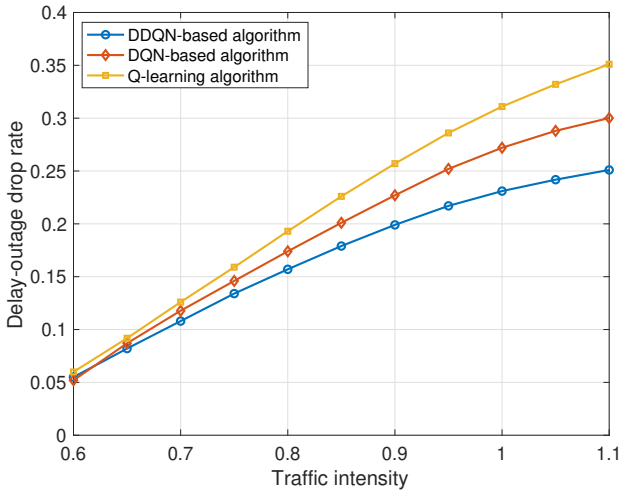


Figure 3.10: Drop rate versus traffic intensity for the overload case.

algorithm.

Figure 3.9 depicts the average delay achieved by the various schemes versus traffic intensity ϱ. Despite the average delay of all four schemes increases as the traffic intensity ϱ increases, the proposed joint scheduling scheme always achieves the lowest delay. In addition, we note that only our proposed scheme and scheme 1 show the trend of convergence when ϱ exceeds the capacity region, while the others diverge. The reason is that the packet-selection process becomes more crucial for better control of packets with different delay requirements when ϱ is large. The comparisons between the proposed joint scheduling scheme and scheme 1, and scheme 2 and scheme 3 also show the importance of admission control. In addition, we conclude that the packet-selection process plays an essential role in reducing average delay by comparing our proposed scheme and scheme 2. Overall, our scheme achieves a delay reduction up to 30%.

Delay outage drop performance

Since the proposed model guarantees per-packet delay by dropping delay outage packets, minimizing the drop rate is one of the key metrics in evaluating the performance. Figure 3.10 plots the delay outage drop rate achieved by the three algorithms versus

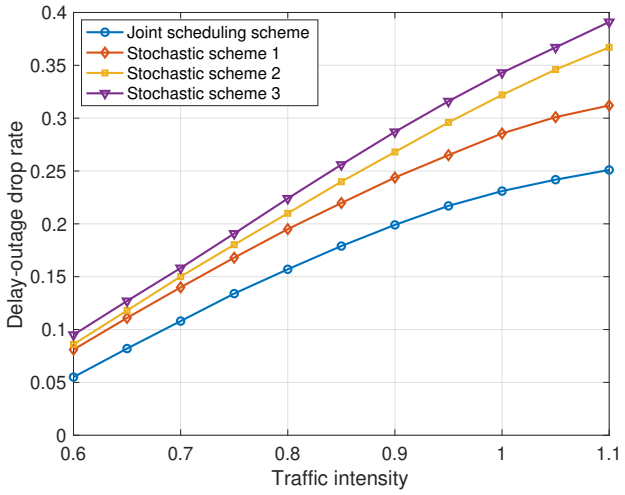
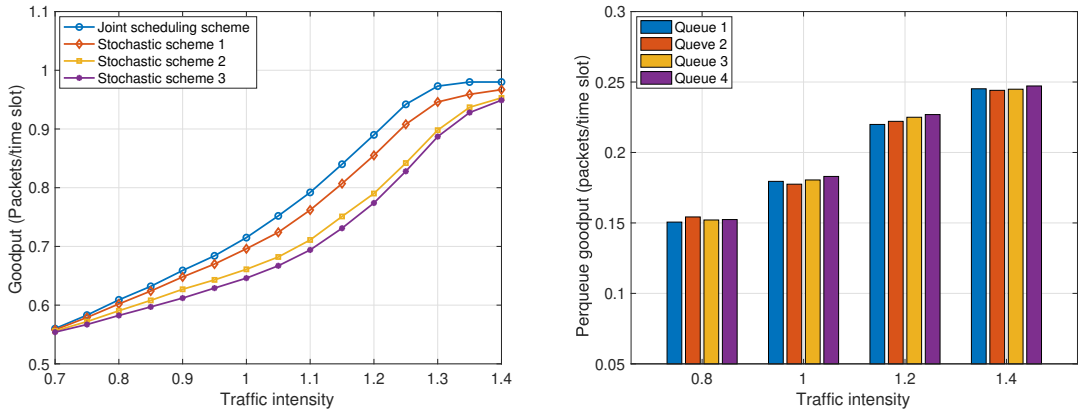


Figure 3.11: Drop rate versus traffic intensity for different design schemes.

traffic intensity ϱ in the case where the arrival rate is outside of the capacity region. It is obvious that the drop rate increases for the three algorithms as ϱ becomes large. The reason is that high traffic intensity leads to longer queue lengths where packets with high delay requirements are more likely to be dropped. On the other hand, the proposed DDQN-based algorithm achieves a much slower increase in terms of drop rate, i.e., a better scheduling policy is achieved. In spite of the increase, all algorithms show the trend of convergence when $\varrho > 1$ thanks to the adoption of the packet-selection process which avoids unnecessary delay outage drops within each queue.

Next, we compare the drop rate performance of our proposed scheduling model with the three benchmark schemes as shown in Figure 3.11. The proposed joint scheduling achieves a significantly lower drop rate up to 60% than the other three schemes. In addition, we observe that the growth trend for our proposed scheme and scheme 1 in Figure 3.11 is similar to that in Figure 3.9. This is because the packet-selection process allows as many packets as possible to be transmitted before reaching their maximum tolerable queuing time. The increasing gap between the proposed joint design scheme and schemes 2 and 3 also indicates the significance of the packet-selection process in our design.



(a) Goodput of different schemes.

(b) Goodput of each queue.

Figure 3.12: Goodput performance under different traffic intensity.

Throughput performance

As the results presented in the previous theoretical analysis, our proposed algorithm can achieve utility maximization while guaranteeing per-packet delay. Here, we consider goodput, i.e., the number of successfully received packets over the total measured time period.

In Figure 3.12 (a), the goodput achieved by the four schemes versus traffic intensity ρ is plotted. As expected, the goodput increases for the four schemes as ρ becomes large and eventually tends to 1 when ρ is sufficiently large. For different values of ρ, our proposed scheme always achieves the highest goodput, while scheme 3 has the lowest goodput. In addition, we observe that the curve of scheme 1 is closer to the curve of our proposed scheme than that of scheme 2. These results further indicate that packet transfer control is more effective than admission control in improving goodput.

In order to demonstrate the fairness of the proposed solution among users, Figure 3.12 (b) plots the goodput per queue versus traffic intensity ϱ for the outside capacity region case. It can be seen that the goodput of all the queues increases as ϱ becomes large. For different values of ϱ, the total goodput of the four queues is equal to the goodput corresponding to the blue curve in Figure 3.12 (a). It can also be seen

that the values of the four queues are very close to each other, which confirms that our formulated model can achieve fairness among users.

3.7 Conclusions

In this chapter, we investigated the resource scheduling problem for time-critical applications with different delay requirements in a single-hop downlink network. To address the problem, a delay-aware selective scheduling scheme based on delay laxity was proposed, in which we particularly considered the drop of packet due to excessive delay. In this context, we formulate an MOOP that minimizes the average queue length while maximizing the average output gain under the constraints of guaranteeing per-packet delay and achieving fairness among users. To cope with the uncertainties in wireless networks, e.g., packet arrival rate and channel condition, we transformed the problem into an MDP and proposed a DDQN-based algorithm to solve it. Simulation results clearly show the superiority of the proposed scheduling scheme and algorithm in reducing delay while improving throughput over other solutions.

While the selective scheduling design proposed in this chapter opens a new perspective of optimizing network utility, many other issues in practice, e.g., wireless link dynamics and user mobility are not considered in the current work. In future work, more effort can be addressed on incorporating environmental dynamics to improve the scalability of our solution.

Chapter 4

QoS-guaranteed Routing Protocol for Intelligent Transportation System: A Hybrid Approach

In Chapter 2 and Chapter 3, we focus on static networks where mobility is ignored. However, the emerging new applications are designed under the context of mobile networks. The network mobility further introduces greater challenges to guaranteed QoS. Furthermore, some applications require information exchange among multiple end-users, for example, enhanced vehicle-to-everything (V2X) communications. Unlike conventional V2X applications such as safety message dissemination which is used for low-level automation with low bandwidth and quality-of-service (QoS) requirements, the emerging new V2X applications are expected to bring advanced driving experience by leveraging various environmental information for real-time autonomous control, which require diverse and stringent QoS support under high mobility [1, 39].

Therefore, in this chapter, we investigate the scheduling and routing problem in vehicular networks. Particularly, we focus on the extended sensor sharing (ESS) application which requires frequent information dissemination in a coverage with guaranteed quality-of-service (QoS) [105]. The frequently used symbols in this chapter is

summarized in Table 4.1.

Table 4.1: Notations and definitions

Symbol	Definitions
u_i	Member vehicle i
$l_{u_i},\ v_{u_i}$	Location and speed of u_i
$\mathcal{U}$	Sets of member vehicles
$\mathcal{M}$	Sets of mobility information
d_{u_i}	Inter-vehicle distance of u_i
$\mathcal{Q}$	Sets of QoS requirements
$B_w,\ B_v$	Bandwidth allocated for V2I and V2V links
$P_w,\ P_v$	Transmission power for V2I link and V2V links
$R_w,\ R_v$	Theoretical link capacity for V2I link and V2V links
$d_{\min},\ d_{\max}$	Derived transmission distance lower and upper bounds
$\mathcal{C}$	Sets of clusters
$\boldsymbol{CH}$	Sets of cluster heads
G_k	Trellis graph representation of cluster k
$\mathcal{P}_k$	Routing path for cluster k
GPCA	Global protocol control agent
LPCA	Local protocol control agent

4.1 Introduction

ESS is difficult to support due to the following reasons. First, ESS requires a wide range of QoS support for transmitting various types of data, e.g., it requires $3 \sim 100$ ms end-to-end (E2E) latency, $10 \sim 1000$ Mbps data rate, $90 \sim 99.99$ % reliability, and $10 \sim 100$ meters coverage [114]. Second, high mobility causes frequent changes in network topology and channel variations, which further introduces higher topology control overhead, longer E2E delay, and higher link failure probability [15, 61]. Third, sharing the same data among multiple end-users requires high resource efficiency. Due to the shared nature of the wireless medium, high interference

impedes network performance [28, 138]. Last but not least, as shared information is required by multiple end users within a certain range, using unicast (one-to-one) service mode is uneconomical while broadcast (one-to-all) mode causes high information redundancy in the network [6].

Vehicles and roadside infrastructure involved in the ESS system naturally form a network. An efficient routing algorithm is desired to enable data sharing with guaranteed QoS for multiple end-users subscribed to the same ESS session. While there are few works proposed specifically for ESS applications, in the past decades, many routing solutions have been proposed for general V2X applications [13, 122]. A delay-aware grid-based geographic routing (DGGR) protocol was proposed in [29] to address the local maxima and data congestion issues in a distributed manner. The target region was first divided into grid zones, based on which, a road weight evaluation (RWE) algorithm was developed to determine the routing path among the grid zones. More recently, the software-defined vehicular network (SDVN) has been introduced, which integrates the concept of software-defined network (SDN) into vehicular networks for higher flexibility and programmability [62, 134]. Several routing algorithms were developed based on the SDVN architecture. [109] proposed a centralized cluster-based routing algorithm to trade-off between bandwidth cost and E2E delay. In [109], both vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications were considered to find the optimal routing strategy.

However, these solutions are insufficient to support ESS. First, a fully centralized control solution requires frequent topology updates to maintain performance leading to high control overhead. Second, centralized routing protocols cannot quickly respond to short-term network changes due to mobility. Third, it is very difficult if not possible for distributed routing solutions to guarantee QoS requirements due to a lack of global coordination.

To bridge the gap, we present a QoS-guaranteed clustering and routing protocol (QCRP) following the SET protocol architecture we proposed in our previous work [21]. A protocol control agent (PCA) is designed to leverage global and local

network information to make control decisions[1]. It uses the global network information to find the optimal routing path to satisfy the E2E QoS requirements and the local network information to fine-tune control decisions to adapt to network dynamics. Moreover, topology control and re-routing are adopted to reduce control overhead and maintain network performance over time.

Specifically, we first compute the inter-vehicle reference distances based on the QoS requirements, which yield the upper and lower bounds for clustering and path calculation. Next, the network is clustered by the reference distance to ensure network connectivity. To mitigate the impact of network disconnection caused by network mobility, a proactive clustering algorithm based on the Long-Short-Term-Memory (LSTM) is adopted in QCRP. GPCA, i.e., the PCA with global network information, predicts the number of time slots before the network disconnection occurs, i.e., the inter-vehicle distance exceeds the reference distance upper bound, using historical vehicle mobility traces. GPCA proactively determines the global topology update period based on the prediction results. For route planning, we consider a hybrid usage of V2V and V2I links for intra and inter-cluster communications. A QoS-guaranteed routing algorithm (QGR) is developed to find the optimal routing path within each cluster. Since topology variations may lead to network performance degradation, a distributed re-routing algorithm is developed. Within the global topology update period, the LPCA deployed at each relay vehicle along the original routing path re-selects the next hop based on the residual link lifetime estimated using local network observations. In this case, the routing paths are updated timely without the need for global topology updates and path calculations to maintain the required QoS.

The rest of the chapter is organized as follows. We introduce the related work in Section 4.2. In Section 4.3, we explain the proposed network architecture and system model. Next, we formulate our problems in Section 4.4 and propose the corresponding solutions in Section 4.5. In Section 4.6, we evaluate our work with extensive simulations followed by conclusions and further research issues in Section 4.7.

[1]Thus, two types of PCA, global PCA (GPCA) and local PCA (LPCA) are defined in QCRP.

4.2 Related Work

4.2.1 Distributed Solutions

The routing protocols that compute routing paths in a distributed fashion without the need for a global controller are referred to as distributed solutions. Distributed solutions have the benefits of low overhead and high adaptiveness to topology changes but suffer from frequent link outage issues in highly mobile environments [46]. To address this issue, many solutions have been proposed in the literature. They can be divided into several sub-categories by their methods. Location-based methods adopt geographic locations for route selection. Most location-based methods are developed based on the Greedy Perimeter Stateless Routing (GPSR) protocol [67, 122], for example, an enhanced GPSR protocol was proposed in [17] to reduce link outage and improve throughput by stabilizing the routing path. A connectivity-aware routing (CAR) protocol [92] was proposed to address the high delay issue caused by the carry-and-forward mechanism, where the route with less network disconnection probability was selected for packet transmission. Broadcast-based solutions improve network performance by suppressing the broadcast messages in data dissemination. In [128], a distributed vehicular broadcast (DV-CAST) protocol relying only on the local topology was proposed to address the broadcast storm and low connectivity issues with a small overhead. Later, in [68], a velocity and position-based broadcast suppression protocol (VP-CAST) was proposed for VANETs. The infrastructure-based methods utilize roadside infrastructures to improve network connectivity while satisfying QoS requirements [104].

4.2.2 Centralized Solutions

Centralized solutions require the knowledge of global network topology before route calculation. Conventional routing protocols such as Optimized Link State Routing Protocol (OLSR) and its variants are initially designed for mobile ad hoc networks

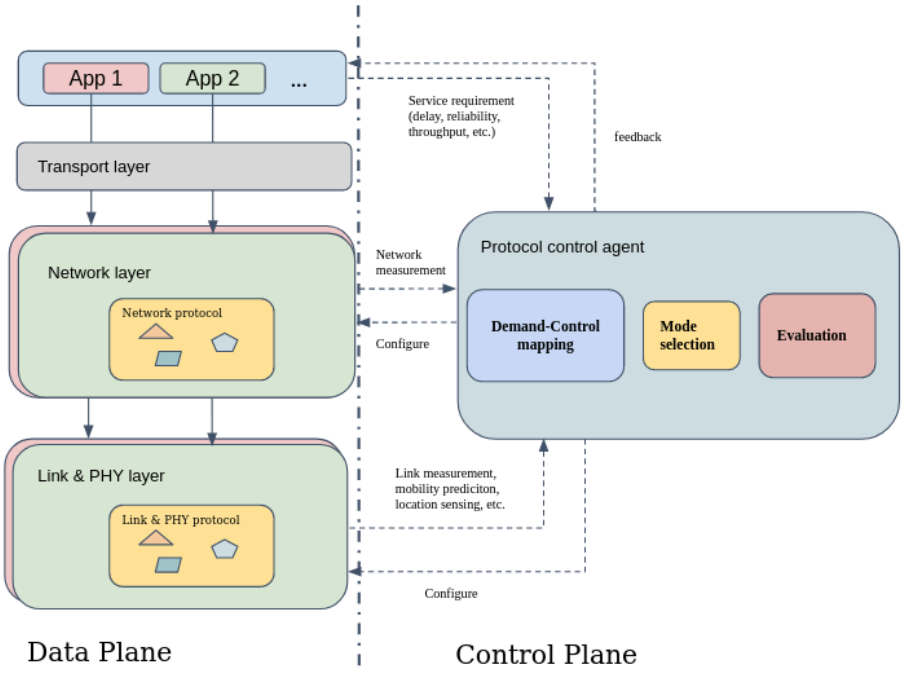
and later extended to vehicular networks [32,42,129]. However, they suffer from high control overhead for topology updates and long convergence delay [119]. Later, SDN attracted great attention which decouples the control plane and data plane. A new paradigm named SDVN leveraging SDN in vehicular networks has been explored in literature [18,135] where the SDN controller is deployed at the edge cloud or road-side units (RSUs) for centralized route calculation and faster response to topology changes. More recently, with the development of edge computing and learning algorithms, researchers propose to adopt learning algorithms at mobile edge [87,141,142]. In [124], an artificial neural network-based vehicle mobility prediction model was developed to assist route calculation at the SDN controller. [76] proposed a greedy routing algorithm based on Graph Convolutional Network using the SDVN architecture to improve packet delivery ratio and delay.

While these works have shown promising performance in certain use cases, they are insufficient for ESS. The distributed solutions employ a best-effort strategy for path discovery and fail to guarantee any QoS requirements due to a lack of global network knowledge. The centralized solutions require global knowledge for path calculation consuming high control overhead and time to synchronize and update frequently to maintain the network performance. The cost can grow exponentially with the increase of ESS subscribers and network size which are uneconomical in practice. Therefore, a scalable, adaptive, and low cost routing protocol is desired to bridge the gap.

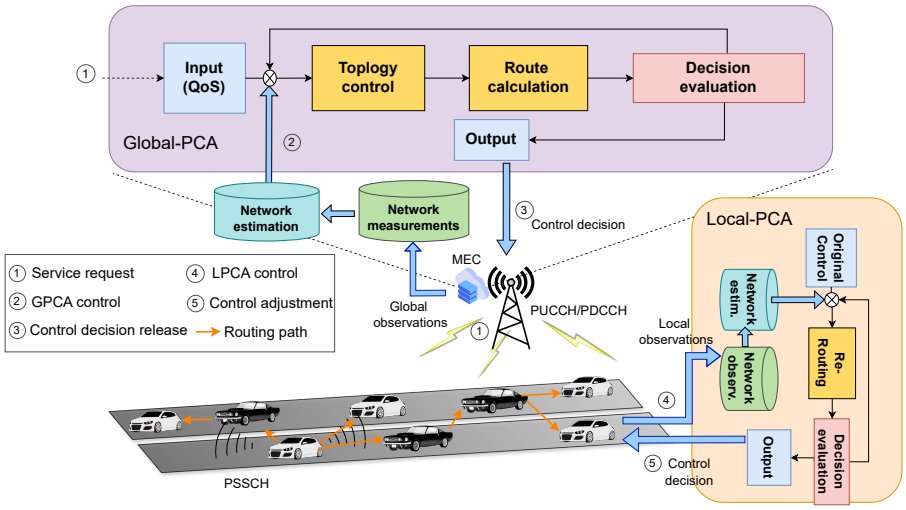
4.3 System Model

4.3.1 Network Architecture

We develop QCRP following the SET protocol architecture [21], where the control decisions are generated using cross-layer information. Each network entity deploys a PCA to configure protocols based on the observed network information and application requirements as shown in Fig. 4.1 (a). For each packet/flow with specific QoS



(a) QCRP protocol architecture



(b) Network architecture for ESS applications in vehicular networks

Figure 4.1: An overview of network architecture.

requirements (e.g., delay, data rate, and reliability), PCA determines a control decision based on the current network conditions. In addition, PCA periodically evaluates and updates its control strategy to maintain QoS performance and reports errors to the application if the network cannot support the requirements.

We show the proposed network architecture in Fig. 4.1 (b) where two types of PCAs are defined, namely, GPCA and LPCA, to collect and respond to global and local network information, respectively. While the global network knowledge is essential for computing the optimal routing paths, it causes high overhead and delay to update, especially in vehicular networks, due to high network dynamics. On the other hand, local network knowledge reflects the network conditions in real time and is less costly to update in the neighbourhood. GPCA is deployed at the central controller to determine the optimal routing path for guaranteeing the E2E service requirements using the global network knowledge. LPCA is deployed in each network entity along the routing path which is responsible for adjusting GPCA's control decisions using real-time local network observations.

4.3.2 Assumptions and Network Settings

In our scenario, we refer to all vehicles that subscribed to one ESS session as member vehicles and those selected to relay data packets as relay vehicles. The source vehicle is also considered a relay vehicle. We assume each vehicle has a unique identification number and periodically uploads its mobility information (e.g., locations, speed, link conditions, etc) to the base station (BS) via the physical uplink control channel (PUCCH) to construct a global network topology. GPCA is deployed at the BS side. We define five stages in a complete E2E data transmission process in the proposed network architecture, namely, service request, global control, control decision release, local control, and control adjustment as shown in the right figure of Fig. 4.1.

In the first stage, the ESS application initializes the service by sending its QoS requirements[2] to the BS. Next, given the global network topology and ESS service requirements, GPCA computes the optimal control decisions, i.e., global network topology update period and routing path, in the second stage and broadcasts it to the network using the physical downlink control channel (PDCCH) in the third stage. While the source vehicle starts data transmission after the third stage, the

[2]For example, E2E delay, data rate, reliability, coverage, etc.

LPCA deployed at each relay vehicle performs local control to maintain the network performance by collecting the mobility and link information of its neighbours and adjusting the routing paths when link breakage occurs as indicated by stage four and five. Similar to [109, 124], we assume all vehicles are equipped with transceivers for V2V and V2I communications using the physical sidelink shared channel (PSSCH) and physical uplink shared channel (PUSCH) and physical downlink shared channel (PDSCH) for data transmission, respectively. Member vehicles periodically broadcast their mobility information to their one-hop relay vehicles via beacon messages using the physical sidelink control channel (PSCCH). This setting addresses frequent topology variations via V2V communications and ensures coverage via V2I communications.

4.3.3 Network Model

We consider a highway scenario as depicted in the right side of Fig. 4.1, where vehicles traverse a high-speed highway and fall within the coverage range of a solitary BS. Our analysis specifically focuses on a single source case, wherein only one vehicle shares data with other vehicles within the network.

Denote $\mathcal{U}$ as the complete set of member vehicles (we use vehicles for short in the rest of the chapter). In total, there are U vehicles. Each vehicle uploads its mobility information denoted by $M_{u_i} = \{l_{u_i}, v_{u_i}\},\ u_i \in \mathcal{U}$ to the BS, where $l_{u_i} = (x_{u_i}, y_{u_i})$ is the location of vehicle u_i and v_{u_i} is its speed. Thus, the global network knowledge can be represented as $\mathcal{M} = \{M_{u_1}, M_{u_2}, \cdots, M_{u_N}\}$. On receiving $\mathcal{M}$, GPCA computes the inter-vehicle distance:

$$d_{u_i} = \sqrt{(x_{u_i} - x_{u_j})^2 + (y_{u_i} - y_{u_j})^2},\ u_i, u_j \in \mathcal{U}. \tag{4.1}$$

In addition, we denote the QoS requirements by a tuple $\mathcal{Q} = \{\theta_{\text{rel}}, \theta_{\text{rate}}, \theta_{\text{del}}, \theta_{\text{cov}}\}$, where θ_{rel} is the E2E reliability requirement, θ_{rate} is the data rate requirement, θ_{del} is the E2E delay requirement, and θ_{cov} is the coverage requirement. $\mathcal{Q}$ is known by all

vehicles and the same for each member vehicle subscribing for the same ESS session.

4.3.4 Channel Model

The uplink and downlink average data rate of the V2I link can be estimated by

$$R_w = \eta \cdot B_w \cdot \log_2 \left(1 + \frac{P_w}{N_0} \delta d_w^{-\alpha} |h|^2\right), \ w \in \{\text{up}, \text{dw}\}, \tag{4.2}$$

where $\eta \in (0, 1]$ is a constant coefficient determined by the transceiver hardware efficiency, B_w is the V2I link bandwidth, P_w is the transmission power, d_w is the uplink/downlink transmission distance, δ is the shadowing component, h is the Rayleigh-distributed fading coefficient with $\mathbb{E}(|h|^2) = 1$, N_0 is the power of additive white Gaussian noise (AWGN), and α is the path loss component. Similarly, the average data rate of a V2V link can be estimated:

$$R_v = \eta \cdot B_v \cdot \log_2 \left(1 + \frac{P_v}{N_0} \delta d_v^{-\alpha} |h|^2\right), \tag{4.3}$$

where B_v, P_v, and d_v are the V2V communication bandwidth, transmission power, and inter-vehicle distance, respectively. Given the data rate requirement θ_{rate}, the transmission distance constraint on data rate for V2V communication can be derived as follows[3]

$$d_{\text{rate}} \leq ((2^{\frac{\theta_{\text{rate}}}{\eta B_v}} - 1) \cdot \frac{N_0}{P_v \delta |h|^2})^{-\frac{1}{\alpha}}. \tag{4.4}$$

i.e., the transmission distance d_v can be d_{rate} at maximum to satisfy the data rate requirement.

[3]Note that in practical systems, the above channel models may not be accurate due to randomness. In our design, the channel model is used by the GPCA to estimate link data rates and perform route planning.

4.3.5 Delay Model

There are mainly five delay components at each hop, namely, transmission delay T_t, medium access delay (or contention delay) T_c, queuing delay T_q, propagation delay, and processing delay, where the propagation and processing delay are negligible (at the level of a few microseconds). In this case, the E2E time cost of a path $\mathcal{P}$ can be calculated as

$$tc_{\mathcal{P}} = \sum_{i \in \mathcal{P}} tc_i = \sum_{i \in \mathcal{P}} T_t^i + T_c^i + T_q^i, \tag{4.5}$$

where tc_i refers to the time cost at each hop i along the path. Since we consider time-sensitive packet transmission, we assume the sum of medium access delay and queuing delay can be upper bounded by a constant value τ. The transmission delay can be computed by $T_t = L/R_v^i$, where L is the packet size and R_v^i is the data rate of the i-th hop[4]. Thus, the E2E time cost of a path $\mathcal{P}$ can be rewritten as

$$tc_{\mathcal{P}} = \sum_{i \in \mathcal{P}} (\frac{L}{R_v^i} + \tau_i). \tag{4.6}$$

Furthermore, given the E2E delay, coverage, and data rate requirements, the transmission distance constraint in terms of time cost is given by

$$d_{\text{del}} = Dist \times \frac{L + \tau_{\max} \cdot \theta_{\text{rate}}}{\theta_{\text{del}} \cdot \theta_{\text{rate}}}, \tag{4.7}$$

where $Dist$ is the distance from the source to the furthest member vehicle and $\tau_{\max}$ is the maximum delay constant introduced by queuing delay and contention delay. In other words, the transmission distance d_v is lower bounded by d_{del} to satisfy the delay requirement.

[4]The delay model can be extended by considering variable per-hop delay as shown in [140].

4.3.6 Connectivity Model

Since wireless links are highly dynamic, analyzing the connectivity of each link is fundamental to achieving the QoS requirements. We use the packet delivery ratio (PDR) of a link to characterize the per-hop transmission reliability

$$P_e = (1 - p_b)^L, \tag{4.8}$$

where p_b is the bit error rate (BER)[5]. Similarly, the E2E reliability P_s can be derived as follows,

$$P_s = \prod_{i \in \mathcal{P}} P_e^i = \prod_{i \in \mathcal{P}} ((1 - p_b)^L)^i \geq \theta_{\text{rel}}. \tag{4.9}$$

Since p_b is determined by the modulation scheme and signal-to-noise ratio (SNR), it can be represented by a function of distance, i.e., $p_b = f(d)$[6]. Thus, the transmission distance constraint on reliability can be derived as follows

$$(1 - f(d))^L \geq \theta_{\text{rel}}. \tag{4.10}$$

d_{rel} denotes the solution to (4.10) when the equality sign holds.

In summary, we define d_{del} as the minimum reference distance $d_{\min}$ and $d_{\max} = \min(d_{\text{rate}}, d_{\text{rel}})$ as the maximum reference distance to facilitate topology control which characterizes the lower and upper bound of a transmission range. In this context, GPCA and LPCA determine whether the current routing plan can support the QoS requirements by comparing the distance between two relay candidates with $d_{\min}$ and $d_{\max}$. An error message will be sent to the ESS application for adjusting QoS requirement when $d_{\min} > d_{\max}$, i.e., more E2E delay time budget should be given to ensure a guaranteed performance. If there exists a non-empty relay candidate set that satisfies the transmission range, a routing path is calculated by selecting relay vehicles from

[5]BER can be improved using physical-layer error correction code (ECC) such as forward error correction (FEC) [136], etc.

[6]We use the BPSK modulation scheme as an example to derive this function as shown in the Appendix.

the candidate set.

4.4 Problem Formulation

4.4.1 Network Clustering

Since not all vehicles in the network $\mathcal{U}$ can be reached using V2V links due to the transmission distance constraints imposed by the QoS requirements, GPCA divides the network into clusters where all vehicles within the same cluster are connected[7]. In this context, we propose a two-step clustering algorithm as summarized in Algorithm 6 to guarantee the connectivity of the vehicular network.

In the first step (line:3-9), GPCA computes the inter-vehicle distance $d_{u_i}(t)$ for each $u_i \in \mathcal{U}$ at time t and estimates the future inter-vehicle distance after t_0 time slots, i.e., $\hat{d}_{u_i}[t] = d_{u_i}[t + t_0]\ \forall u_i \in \mathcal{U}$. Next, GPCA divides the network into sub-networks if $\max\{d_{u_i}[t], \hat{d}_{u_i}[t]\} \geq d_{\max}$. This step guarantees the connectivity of each sub-network, where all vehicles within the same sub-network can be reached while satisfying the per-hop QoS requirement. By the end of this step, $\mathcal{U}$ is divided into K_0 sub-networks. In the second step (line:10-18), GPCA further evaluates the coverage distance of each sub-network $\text{cov}(\hat{\mathcal{U}}_k)$ and a sub-network will be clustered again if the E2E time cost exceeds the delay budget:

$$tc_{\max} = \frac{\text{cov}(\hat{\mathcal{U}}_k)}{d_{\min}} \cdot (\frac{L}{\theta_{\text{rate}}} + \tau_{\max}) > \theta_{\text{del}},\ \mathcal{U}_k \in \hat{\boldsymbol{\mathcal{U}}}, \tag{4.11}$$

where $\frac{\text{cov}(\hat{\mathcal{U}}_k)}{d_{\min}}$ computes the maximum number of hops in a sub-network and $\frac{L}{\theta_{\text{rate}}} + \tau_{\max}$ is the maximum per-hop time cost. Thus, a sub-network $\hat{\mathcal{U}}_k$ can be divided into $K = \lceil \frac{\theta_{\text{del}}}{tc_{\max}} \rceil$ clusters. Next, GPCA uniformly divides $\hat{\mathcal{U}}_k$ into K clusters $\mathcal{C}_k = \{C_1, \ldots, C_K\}$

[7]In this chapter, we define a network is connected if there exists a path connecting any pair of vehicles while satisfying the E2E QoS requirements.

Algorithm 6 Clustering algorithm

Input: Network mobility information $\mathcal{M}$ and reference distances $(d_{\min}, d_{\max})$.
Output: Cluster set $\boldsymbol{\mathcal{C}}$ and cluster head set **CH**.
// First-step clustering
Obtain network mobility at time t:
for each u_i in $\mathcal{U}$ **do**
    Compute $d_{u_i}[t]$ and $\hat{d}_{u_i}[t]$
    Split network if $\max\{d_{u_i}[t], \hat{d}_{u_i}[t]\} \geq d_{\max}$
end for
Obtain $\hat{\boldsymbol{\mathcal{U}}} = \{\hat{\mathcal{U}}_1, \hat{\mathcal{U}}_2, \cdots, \hat{\mathcal{U}}_{K_0}\}$
// Second-step clustering
for each $\hat{\mathcal{U}}_k \in \hat{\boldsymbol{\mathcal{U}}}$ **do**
    Compute $\mathrm{cov}(\hat{\mathcal{U}}_k)$ and K
    Uniformly divide $\hat{\mathcal{U}}_k$ into clusters $\mathcal{C}_k = \{C_1, \ldots, C_K\}$.
    for each $C_k \in \mathcal{C}_k$ **do**
        Compute: $\mathrm{CH}_k = \underset{v_j \in C_k}{\arg\min}\ l_{v_j} - \sum_{v_j \in C_k} \frac{l_{v_j}}{|C_k|}$.
    end for
end for

and the vehicle closest to the cluster center is selected as the cluster head (CH), i.e.,

$$\mathrm{CH}_k = \underset{u_j \in C_k}{\arg\min}\ l_{u_j} - \sum_{u_j \in \mathcal{C}_K} \frac{l_{u_j}}{|C_k|}, \tag{4.12}$$

where $|C_k|$ is the total number of vehicles within C_k and $\sum_{u_j \in \mathcal{C}_K} \frac{l_{u_j}}{|C_k|}$ yields the centroid of cluster C_k. Finally, we obtain the cluster set $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ and the CH set $\mathbf{CH} = \{\mathrm{CH}_1, \mathrm{CH}_2, \cdots, \mathrm{CH}_K\}$. In our design, each CH is responsible for receiving data packets from the source vehicle via V2I links and distributing them within its cluster.

However, estimating the inter-vehicle distance in t_0 can be challenging due to the time-varying vehicular mobility patterns, which further affect the connectivity of the network. Therefore, we aim to develop an effective solution to predict the connectivity duration time based on the proposed clustering algorithm and proactively re-cluster the network before the cluster loses its connectivity.

4.4.2 Route Planning

We assume guaranteed QoS performance for V2I links as they use dedicated frequency bands for data transmission in a single hop. As for intra-cluster V2V communications, we focus on a multi-hop scenario considering the transmission distance constraints and coverage requirements.

We consider a single-trip data transmission in our scenario, where the sender only transmits each packet once without packet retransmission. To support the E2E QoS requirements, the available routing paths need to satisfy the following constraints: (i) Guaranteed throughput, i.e., each V2V link should guarantee a minimum data rate of θ_{rate}; (ii) Guaranteed E2E reliability, i.e., the E2E PDR should be greater than θ_{rel}. While ESS applications are time-sensitive, we formulate the routing problem to minimize the E2E delay:

$$\text{P1:} \quad \arg\min_{\mathcal{P}} \quad \sum_{u_i \in \mathcal{P}} \frac{L}{R_v^{(u_i)}} + \tau_{u_i} \tag{4.13a}$$

$$\text{s.t.} \quad R_v = B_v \cdot \log_2 \left(1 + \frac{P_v}{N_0} \delta d_v^{-\alpha} |h|^2\right) \geq \theta_{\text{rate}} \tag{4.13b}$$

$$(P_v)_{\text{dB}} - PL(d_v)_{\text{dB}} - N_{\text{dB}} \geq \theta_{\text{SNR}}, \tag{4.13c}$$

$$(1 - p_b)^L \geq \theta_{\text{rel}}, \tag{4.13d}$$

$$\prod_{i \in \mathcal{P}} ((1 - p_b)^L)^i \geq \theta_{\text{rel}}, . \tag{4.13e}$$

where u_i is the relay vehicle selected at hop i, (4.13b) is the the per-hop data rate constraint, (4.13c) is the link condition constraint, and (4.13d) and (4.13e) are the per-hop and E2E reliability constraints, respectively. The solution to P1 yields the optimal routing path.

4.4.3 In-network Control

We use a simple example as shown in Fig. 4.2 to explain the in-network control performed by each LPCA. After the GPCA broadcasts the control decisions to the

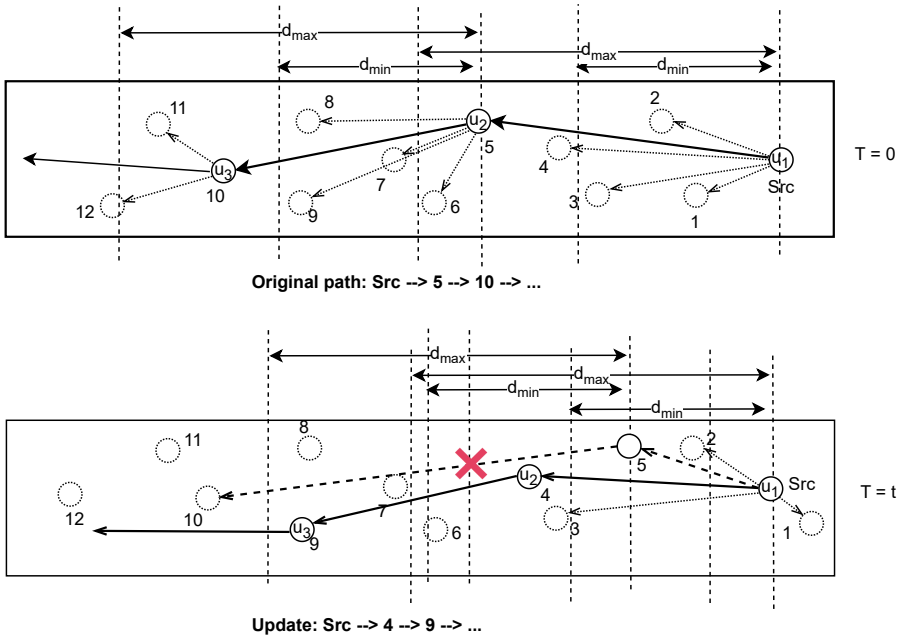


Figure 4.2: In-network control. The original path fails the transmission distance constraints after t time slots due to mobility. LPCA performs re-routing to locally update the routing path.

network, all relay vehicles start data transmission following the control decision. In the meantime, member vehicles periodically send beacon messages containing mobility information to their corresponding relay vehicles after packet reception. In this context, each relay vehicle can construct a one-hop local network topology.

Although GPCA provides optimal control decisions, the network topology changes over time due to mobility, leading to QoS performance degradation. To address this issue, the LPCA at each relay vehicle adjusts the original control decision by selecting a different next-hop relay based on the local network observations. For simplicity, we refer to all vehicles within the transmission range as relay candidates. Such location-based routing strategy is heavily discussed in literature [122]. However, most protocols are developed in a best-effort fashion and are insufficient to guarantee E2E performance due to the lack of global knowledge and QoS requirements. Thus, how to efficiently select each relay candidate to adapt to network dynamics while maintaining the E2E QoS guarantee remains a challenging issue.

4.5 QoS-guaranteed Routing Protocol

4.5.1 Predictive Clustering Algorithm with Feedback Control

We assume MEC is adopted in an ESS-enabled system to execute heavy computation tasks to support GPCA. While vehicle mobility patterns are simple in a highway environment as opposed to urban settings, accurately modelling these patterns in practice remains challenging due to the inherent speed and behavioural diversity of vehicles. Furthermore, the accurate estimation of SNR and consequently, the cluster connectivity is contingent upon precise inter-vehicle distance prediction, taking into account path loss and fading/shadowing effects in the wireless channel. On the other hand, a rich set of vehicle mobility data is uploaded to the BS provisioning the fundamentals for data-driven prediction models. In this context, we propose a proactive clustering algorithm to dynamically determine the cluster update period using LSTM and historical vehicle mobility data[8].

		GPR	LR	SVR	LSTM
Speed	Slow	0.0008	0.0206	0.0225	**0.0001**
	Normal	0.0011	0.0205	0.0225	**0.0001**
	Fast	0.0015	0.0205	0.0227	**0.0001**

Table 4.2: Mean square error (MSE) of four prediction models: Gaussian Process Regression (GPR), Linear Regression (LR), Support Vector Regression (SVR), and LSTM predicting vehicles' locations in 10 time slots with different speed.

An accurate prediction model not only helps to maintain the connectivity of each cluster but also reduces the control overhead between BS and vehicles. However, the prediction accuracy decays as the prediction period increases. While a fixed prediction

[8]We adopt LSTM as a proof-of-concept to show the capability of the data-driven prediction model in QCRP. We compared its prediction performance with other regression models as shown in Table 4.2. More advanced prediction model such as graph attention networks (GATs) [84, 130] can be adopted to improve prediction accuracy. In this work, we adopt a single-layer shallow LSTM model and train offline using historical traffic data.

Algorithm 7 Predictive Clustering with Feedback Control

Obatain $\{\mathcal{C}, \mathbf{CH}\}$ and initialize $t_0, t_0^{\min}, t_0^{\max}$
// Mobility prediction phase
for each vehicle $u_i \in \mathcal{U}$ **do**
  Predict the trajectory $\mathbf{D}_{u_i}$ in t_0 slots:
  $\mathbf{D}_{u_i} = \{l_{u_i}(t), l_{u_i}(t+1), \ldots, l_{u_i}(t+t_0)\}$
end for
// Connectivity evaluation phase
for each cluster $C_k \in \mathcal{C}$ **do**
  if $\exists d_{u_i}(t+t_0') > d_{\max},\ u_i \in \mathcal{U}$ **then**
    Set t_0' as the global upload period
  end if
end for
Broadcast $\{\mathcal{C}, \mathbf{CH}, t_0'\}$ to the network.
// Feedback control phase
if receive *error* message **then**
  $t_0 = \min\{t_0/2, t_0^{\min}\}$
else
  $t_0 = \max\{t_0 + t_0'', t_0^{\max}\}$
end if

period is inadaptive to the prediction performance, we propose a proactive prediction period control algorithm following the well-known Additive-Increase-Multiplicative-Decrease (AIMD) to dynamically adjust the prediction period based on the prediction results.

As shown in Algorithm 7, GPCA performs clustering at time t and predicts the mobility pattern of the network in t_0 time slots using LSTM. LSTM essentially predicts the trajectory of each vehicle in the network in time sequence, denoted by $\mathbf{D}_{u_i} = \{l_{u_i}(t), l_{u_i}(t+1), \ldots, l_{u_i}(t+t_0), \},\ \forall u_i \in \mathcal{U}$. Next, GPCA evaluates the connectivity of each cluster using the prediction results by calculating the inter-vehicle distance. A cluster is considered as disconnected if $\forall d_{u_i}(t+t_0') \geq d_{\max}$ at t_0', and $t_0' \leq t_0$. We then adopt t_0' as the upload period when all the member vehicles should upload their mobility data to the BS. Finally, the clustering results $\{\mathcal{C}, \mathbf{CH}, t_0'\}$ are broadcast to the network. After each upload period, GPCA extends its prediction period by t_0'' slots, i.e., $t_0 := t_0 + t_0''$, if no error message is reported to the BS due

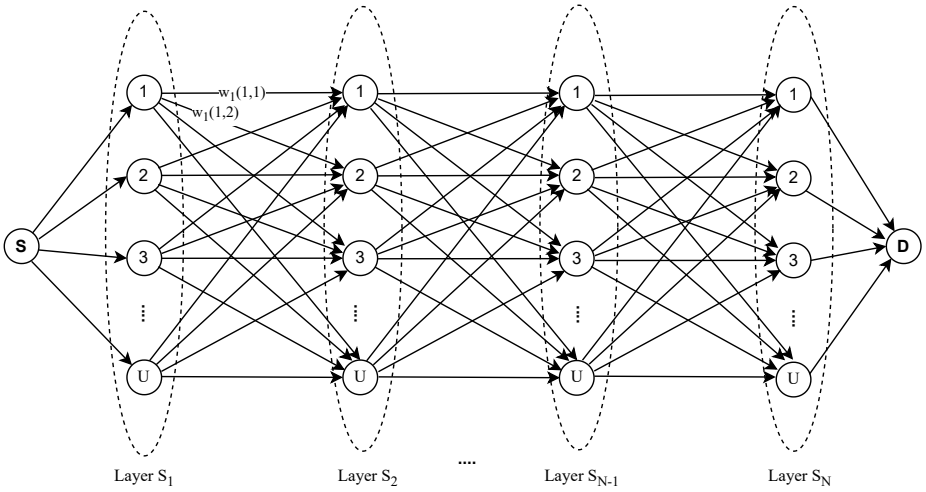


Figure 4.3: Trellis graph representation

to disconnection. Otherwise, t_0 is reduced by half to restore its prediction accuracy. To avoid overwhelming the prediction process, we manually set the lower bound $t_0^{\min}$ and upper bound $t_0^{\max}$ for the prediction period t_0. In summary, the update process can be expressed as follows:

$$t_0 := \begin{cases} \max\{t_0 + t_0'', t_0^{\max}\} & \text{if } \textit{error} \text{ is false,} \\ \min\{t_0/2, t_0^{\min}\} & \text{if } \textit{error} \text{ is true.} \end{cases} \tag{4.14}$$

4.5.2 QoS-guaranteed routing algorithm

To better characterize the routing process, we use the Trellis graph $G =< V, N, E >$ to represent each cluster $\mathcal{C}_k$ as shown in Fig. 4.3. A graph consists of V nodes, E edges, and N layers (representing N hops between the source and destination nodes). Each layer S contains the complete V nodes and all nodes between two layers are fully connected with edges E. We regard the CH of each cluster as the source node and the vehicle at the two ends of each cluster as the destination nodes. Thus, two graphs can be constructed to represent a cluster, i.e., forward G_k^f and backward G_k^b, respectively[9].

[9]However, the trellis graph can grow exponentially with the increased number of vehicles of each cluster, which increases the time complexity of finding the shortest path. To address this issue, we

Next, we explain the proposed QoS-guaranteed routing algorithm to perform route planning for each cluster as summarized in Algorithm 8. Specifically, to incorporate the QoS constraints in our network representation, we define the weight of each edge as follows

$$w_n(i,j) = \log(\bar{p_b}^{(n)} \cdot \bar{tc}^{(n)}),\ \forall i \in S_n, \forall j \in S_{n+1}, \tag{4.15}$$

where $\bar{p_b} = \frac{\theta_{\text{rel}}}{p_{b_{i,j}}}$ and $\bar{tc} = \frac{tc_{i,j}}{\theta_{\text{del}}}$ are the normalized PDR and time cost between node i and node j of two adjacent layers. For instance, $w_1(1,2)$ represents the edge weight from node 1 in S_1 to node 2 in layer S_2. The benefit of such a weight metric design is that both E2E delay and reliability are considered in our graph representation thanks to the sum property of the logarithm. In addition, we set the weight to infinity if the inter-vehicle distance exceeds $d_{\max}$ to remove unsatisfactory links, i.e., $\{w_n(i,j) = \infty | d_{u_i} > d_{\max}\}$. Then, GPCA constructs the corresponding graph G_k for each cluster $C_k \in \mathcal{C}$ (Line:4). In total, the number of paths from the source node to the destination node can be derived $N_{\text{path}} = \prod_{n=1}^{N} |\mathbf{w}_n|$, where $|\mathbf{w}_n|$ is the number of non-infinity-weight edges between two layers in the graph, and there are at most V^N number of paths in a graph. Finally, GPCA computes the shortest path that minimizes the sum of the total weight from the source to the destination for each graph using Dijkstra's algorithm. The routing path $\mathcal{P}$ is then evaluated by the E2E QoS performance metrics before broadcast to the network:

$$tc(\mathcal{P}) = \sum_{n=1}^{N} tc^{(n)},\ P_s(\mathcal{P}) = \prod_{n=1}^{N} P_e^{(n)}.$$

An error message will be reported if the current path fails to support the QoS requirements (Line:5-8), i.e.,

$$error = \begin{cases} 1 & tc(\mathcal{P}) > \theta_{\text{del}} || P_s(\mathcal{P}) < \theta_{\text{rel}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

simplify the Trellis graph by merging multiple nearby vehicles as one node and randomly selecting one of them as the representation of that node.

Algorithm 8 QoS-guaranteed Routing Algorithm

1: **Input:** $\boldsymbol{\mathcal{C}}$, $\boldsymbol{CH}$, $\mathcal{Q}$, $d_{\max}$, and (src, dst).
2: **Output:** Optimal path set $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_K\}$.
3: **for** $\mathcal{C}_k \in \boldsymbol{\mathcal{C}}$ **do**
4: Construct graph $G_k = \{G_k^f, G_k^b\}$, where the edge weight is denoted by

$$w_n(i,j) = \begin{cases} \log(\bar{p}_b \cdot \bar{tc}_{i,j}) & d_{u_i} <= d_{\max}, \\ \infty & du_i > d_{\max}. \end{cases}$$

5: Compute optimal routing path for G_k:
$\mathcal{P}_k = Dijkstra(G_k, src, dst)$.
6: Evaluate tc and P_s for $\mathcal{P}_k$.
7: Report error if $tc^{(k)} > \theta_{\text{del}}$ or $P_s^{(k)} < \theta_{\text{rel}}$.
8: **end for**

In addition, we assume the member vehicles covered between two consecutive relay vehicles can receive the same packet simultaneously during transmission. In this context, QGR essentially yields a tree-like routing path where the CH is the root node, the rely vehicles are child nodes, and the rest member vehicles are leaf nodes.

4.5.3 Residual link lifetime-based relay selection

Despite the routing paths computed by GPCA guaranteeing the required QoS, they fail to continuously maintain the performances over time due to network mobility. To address the issue without more frequent global network topology updates, we propose a distributed relay selection algorithm for the LPCAs deployed at the relay vehicles to perform re-routing when a link is anticipated to break due to increasing transmission distance.

Denote $\mathcal{U}_{\text{ref}}^i$ as the set of vehicles resides within the reference transmission distance of u_i and we refer a vehicle $u_j \in \mathcal{U}_{\text{ref}}^i,\ i \neq j$ as a *relay candidate*. We evaluate the availability of each relay candidate using the residual link lifetime t_{res}^j, which is defined as the time a relay candidate u_j remains within the transmission range of u_i. Denote $d_s,\ s \in \{\text{fwd, bwd}\}$ as the distance of a relay candidate to the front and back edge of the transmission range as shown in Fig. 4.4. Thus, due to the speed differences

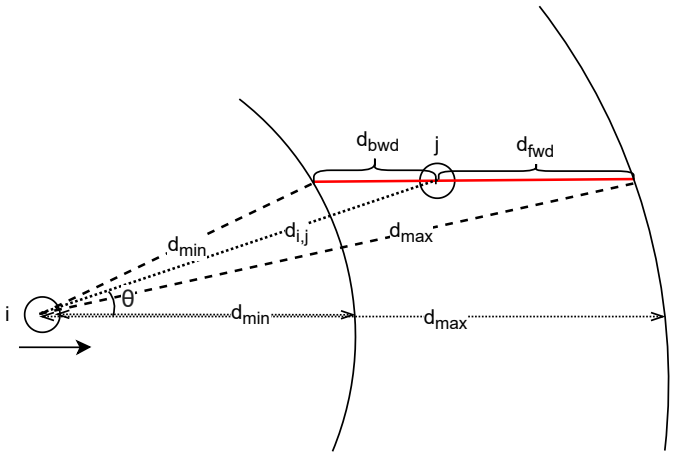


Figure 4.4: Residual link lifetime evaluation

between u_i and u_j and vehicle positions, two types of residual distance can be specified using the law of cosines:

When $v_j > v_i$, the residual distance d_{fwd} satisfies

$$d_{\max}^2 = d_{u_i}^2 + d_{\text{fwd}}^2 - 2d_{u_i}d_{\text{fwd}}\cos(\pi - \theta), \tag{4.17}$$

where θ is the angle between vehicle i and j and $d_{i,j}$ is the inter-vehicle distance. By solving (4.17), we obtain

$$d_{\text{fwd}} = d_{u_i}\cos(\pi - \theta) \pm \sqrt{(d_{\max}^2 - d_{u_i}^2) + (d_{i,j}\cos(\pi - \theta))^2}. \tag{4.18}$$

Here, $\pm$ is determined depending on the sign of d_{fwd}. Similarly, when $v_j < v_i$, the backward residual distance d_{bwd} can be expressed as

$$d_{\text{bwd}} = d_{u_i}\cos\theta \pm \sqrt{(d_{\min}^2 - d_{u_i}^2) + (d_{i,j}\cos\theta)^2}, \tag{4.19}$$

where $d_{\min}$ is the minimum reference distance. Therefore, the residual link lifetime

Algorithm 9 Residual link lifetime-based relay selection

1: **Input:** Local network information $\mathcal{M}_L$, transmission range $\{d_{\min}, d_{\max}\}$, original relay vehicle u_{old}.
2: **Output:** Updated relay vehicle u_{new}
3: Update $T_{\text{budget}} = \theta_{\text{del}} - (t_{\text{start}} - t_{\text{now}})$
4: Update $Dist = Dist - d(l_{u_{new}}, l_{u_{now}})$
5: Update $d_{\min} = Dist \times \frac{L+\tau_{\max}\cdot\theta_{\text{rate}}}{T_{\text{budget}}\cdot\theta_{\text{rate}}}$
6: Compute $t_{\text{res}}^{\text{old}}$, $\bar{t}_{\text{res}}$ using (4.20)
7: **if** $t_{\text{res}}^{\text{old}} < \bar{t}_{\text{res}}$ **then**
8: $\quad u_{new} = \arg\max_{u_{j'} \in \mathcal{U}_{\text{ref}}^i} t_{\text{res}}$
9: **else**
10: $\quad u_{new} = u_{old}$
11: **end if**

can be estimated by

$$t_{\text{res}} = \begin{cases} \frac{d_{\text{fwd}}}{|v_j - v_i|+1} & v_j \geq v_i, \\ \frac{d_{\text{bwd}}}{|v_j - v_i|+1} & v_j < v_i. \end{cases} \tag{4.20}$$

To avoid the oscillation problem, the re-selection procedure is called only if the residual link lifetime of the original relay is less than a threshold[10] and the vehicle with the maximum residual link lifetime is selected as the new relay, i.e.,

$$u_{new} = \begin{cases} \arg\max_{u_{j'} \in \mathcal{U}_{\text{ref}}^i} t_{\text{res}} & t_{\text{res}}^{(u_j)} < \bar{t}_{\text{res}}, \\ u_j & \text{otherwise.} \end{cases} \tag{4.21}$$

It is worth noticing that LPCA updates searching space at each hop by estimating the remaining delay budget T_{budget} and remaining coverage distance $Dist_{\text{rm}}$ as follows

$$T_{\text{budget}} = \theta_{\text{del}} - (t_{\text{start}} - t_{\text{now}}), \tag{4.22}$$

where t_{start} and t_{now} are the timestamps when the data packets are sent out from the

[10] In our implementation, we use the average residual link lifetime of the relay candidate set $\bar{t}_{\text{res}} = \sum_{j \in \mathcal{U}_{\text{ref}}^i} t_{\text{res}_j}$ as the threshold value.

source vehicle and the current reception time, respectively.

$$Dist_{\text{rm}} = \theta_{\text{cov}} - d(l_{u_{\text{now}}}, l_{u_{\text{nxt}}}), \tag{4.23}$$

where $d(l_{u_{\text{now}}}, l_{u_{\text{nxt}}})$ is the distance from the current vehicle to the next relay vehicle. In this context, $d_{\min}$ can be updated by

$$d_{\min} = Dist_{\text{rm}} \times \frac{L + \tau_{\max} \cdot \theta_{\text{rate}}}{T_{\text{budget}} \cdot \theta_{\text{rate}}}. \tag{4.24}$$

The updated $d_{\min}$ is carried in the packet header which will be used for relay selection in the next hop. The complete relay selection algorithm is summarized in Algorithm 9.

In a nutshell, when a data packet arrives at relay vehicle u_i, the corresponding LPCA first updates the remaining delay budget and coverage following (23) and (24) by inspecting the control information carried in the packet header and local network information (Line:3-5). Next, LPCA evaluates the original routing path by computing its residual link lifetime. A new relay vehicle will be selected if $t_{\text{res}}^{\text{old}}$ is smaller than a predefined threshold value (Line:6-11).

4.5.4 Complexity and Control Overhead Analysis

Complexity analysis

The time and space complexities of QCRP mainly come from the centralized route calculation (Algorithm 8). Assuming there are U vehicles divided into K clusters and the average number of vehicles within a cluster $k \in \{1, 2, \ldots, K\}$ is N. Then the space complexity of QCRP is $O(KN)$. Note that since only a small portion of vehicles will be selected as relay vehicles in QCRP, i.e., its graph representation is a sparse matrix, the actual storage is less than $O(KN)$. In terms of time complexity, QCRP first computes the edge weight between two vehicles. Then, it runs the Dijkstra algorithm based on the graph for the shortest calculation. Denote E as the number of activated edges, i.e., edges without infinite weight according to Algorithm 8. Thus,

the time complexity is $O(E + N \log N)$.

Control overhead analysis

QCRP adopts network prediction and local control to reduce the overall control overhead. Here we use a simplified model to illustrate how QCRP reduces the overall control overhead in comparison with a fully centralized solution using real-time global network information. Assume the exchange frequency between vehicles and BS by f_v and f_g, respectively, and the transmission duration is t. Then, the energy cost can be derived as follows

$$E(f, P) = \frac{1}{f} \int_0^t P d\tau,$$

where $f \in \{f_v, f_g\}, P \in \{P_{\text{up}}, P_{\text{dw}}, P_v\}$. Thus, the control overhead of the comparison set is given by

$$E_C = U \cdot E(f_g, P_{\text{up}}) + E(f_g, P_{\text{dw}}).$$

Denote F as the average global topology update period in QCRP and denote N_r as the number of non-relay vehicles. Then, the total energy cost of QCRP is given by

$$\begin{aligned} E_Q &= U \cdot E(\frac{f_g}{F}, P_{\text{up}}) + E(\frac{f_g}{F} P_{\text{dw}}) + N_r \cdot E(f_v, P_v) \\ &= \frac{1}{F} E_C + N_r \cdot E(f_v, P_v). \end{aligned}$$

Furthermore, the local information can be carried in the ACK packet between vehicles. Thus, the control overhead reduction of QCRP can reach up to F times compared to centralized solutions.

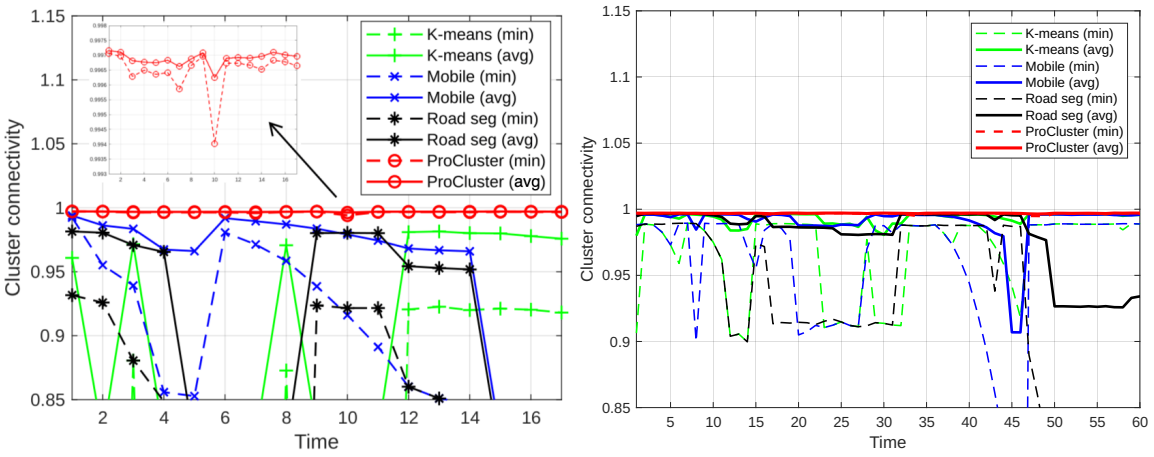
4.6 Performance Evaluation

Table 4.3: Parameters Setting

Parameters	Values
Noise power spectrum density	-100 dBm/Hz
V2V and V2I channel bandwidth B_v, B_w	20 MHz
Transmission power	23 dBm
Carrier frequency	5 GHz
E2E delay requirement θ_{del}	30 ms
Data rate requirement θ_{rate}	80 Mbps
Reliability requirement θ_{rel}	99%
SNR threshold for data transmission θ_{SNR}	23 dB
Cluster coverage θ_{cov}	500 m

4.6.1 Simulation setting

We used the Simulation of Urban MObility (SUMO) simulator to generate four mobility traces with different traffic densities (high, medium, low, and sparse) on a four-lane 5-km highway, where $463, 336, 219$, and 136 vehicles are injected into the system. The BS is deployed in the middle of the road covering the target region. Three types of vehicles with different speed ranges: $[25, 30]$ m/s, $[30, 35]$ m/s, and $[35, 40]$ m/s are considered. All vehicles adopt the Krauss car-following model and lane-change model [121] by default to simulate real car movements. We set all vehicles with the same transmission power of 23 dBm and assign a total bandwidth of 20 MHz in the 5 GHz frequency band for V2V data transmission. As for the QoS requirements, we set the E2E delay, data rate, reliability, and coverage requirements to be 30 ms, 80 Mbps, 99%, and 500 meters, respectively. We select one vehicle in the trace as the source vehicle and set the rest as member vehicles. The source vehicle starts transmission when it enters the target region and we evaluate the network performance over time until it leaves the target region. The complete simulation settings are summarized in Table 4.3.



(a) Short-term performance (b) Long term performance

Figure 4.5: Clustering performance comparison under high traffic density scenario.

4.6.2 Clustering algorithm performance verification

First, we evaluate the connectivity performance of our proposed clustering algorithm, denoted by ProCluster, and compare it with two other clustering algorithms, namely, K-means and Road segmentation[11]. The connectivity of each cluster is evaluated using the connectivity model we derived in (4.9). In addition, we also evaluate the performance Algorithm 6 (denoted by Mobile) to highlight the effectiveness of mobility prediction in ProCluster.

In our simulation, we first evaluate all algorithms in a short period without updating the network topology. We use both the average network connectivity, i.e., average connectivity measurements of all clusters denoted in solid lines, and the minimal network connectivity, i.e., the cluster with the minimum connectivity denoted in dashed lines, to characterize the overall performance of each clustering algorithm. We use the high-traffic density scenario as an example. As shown in Fig. 4.5(a) ProCluster performed the best maintaining high connectivity throughout the experiment

[11]For K-means algorithm, we determine the number of cluster heads by dividing the network size, i.e., the distance between the front-edge vehicle and the back-edge vehicle, by the coverage requirement. For road segmentation (Road seg), we employ the approach adopted in [109] by uniformly clustering the target road into equal-length segments and selecting the centroid vehicle as the cluster head.

Table 4.4: Connectivity performance comparison

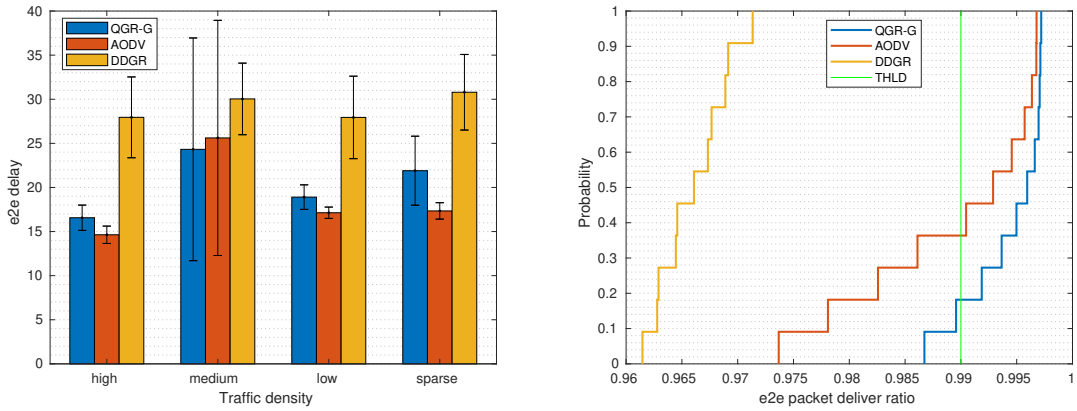
Algorithms	High	Medium	Low	Sparse
ProCluster	**0.9974**	**0.9976**	**0.9969**	**0.9973**
	0.9967	**0.9968**	**0.9965**	**0.9959**
Mobile	0.9877	0.9872	0.9901	0.9911
	0.9271	0.8551	0.9379	0.9638
K-means	0.9871	0.9829	0.9924	0.9911
	0.9150	0.8096	0.9625	0.9638
Road seg	0.9212	0.9627	0.9796	0.9854
	0.6428	0.6776	0.8040	0.8861

with tolerable fluctuations. The connectivity performance using Mobile performed the second best with its average connectivity satisfying the requirement most of the time thanks to its two-step cluster design, however, its minimal connectivity failed the requirement due to network mobility. On the other hand, K-means and Road seg performed worse with high fluctuation in their connectivity performances as they only depend on current topology and are unaware of network mobility.

Next, we fix the topology update period at 10 seconds to verify their long-term performance as shown in Fig. 4.5(b). Similarly, ProCluster maintained the network cluster connectivity throughout the simulation period while the rest restored their performance only after each update. The network is updated adaptively based on the prediction of vehicle mobility. Finally, we summarize the average and minimal connectivity performances under different traffic densities in Table 4.4. Overall, ProCluster achieves an improvement up to 55% compared to other clustering algorithms particularly when traffic density is high. This is because fewer clusters are formed in high-density networks causing higher link outage probability at the edge of each cluster due to vehicle mobility.

4.6.3 Global route planning performance verification

In this experiment, we evaluate the effectiveness of our proposed global routing algorithm employed by GPCA, i.e., the QGR algorithm without LPCA's local control (denoted by QGR-G), in guaranteeing the E2E delay and reliability. Specifically, we



(a) Average E2E delay comparison

(b) Average E2E PDR comparison

Figure 4.6: Global routing algorithm comparison without local control.

compare the E2E delay and reliability performance of QGR-G with the DGGR algorithm proposed in [29] and use AODV as the benchmark[12]. In our evaluation, we first compute the routing path for each cluster at time $t = 0$ using the mobility traces generated by SUMO and compute the theoretical average E2E delay and reliability performances of the entire network over one update period (i.e., the same routing paths are used for data transmission during this period). As shown in Fig. 4.6(a), both AODV and QGR-G compute routing paths satisfying the E2E delay under different traffic densities. AODV provided a slightly better delay performance in some cases, compared to QGR-G as it essentially computes the minimum-hop paths. On the other hand, DGGR fails to satisfy the delay performance as it requires more V2I links for data transmission when the inter-vehicle distance increases and lack of global network topology. Note that the E2E delay can easily reach up to a few seconds for the original DGGR algorithm based on the carry-and-forward mechanism.

As for the E2E reliability performance, we use the Dense traffic scenario as an example. As shown in Fig. 4.6(b), QGR-G performed the best achieving a 0.8 probability that satisfies the reliability requirement (99%) while AODV only provides a 0.6 probability guarantee. This is because QGR jointly considered the reliability and

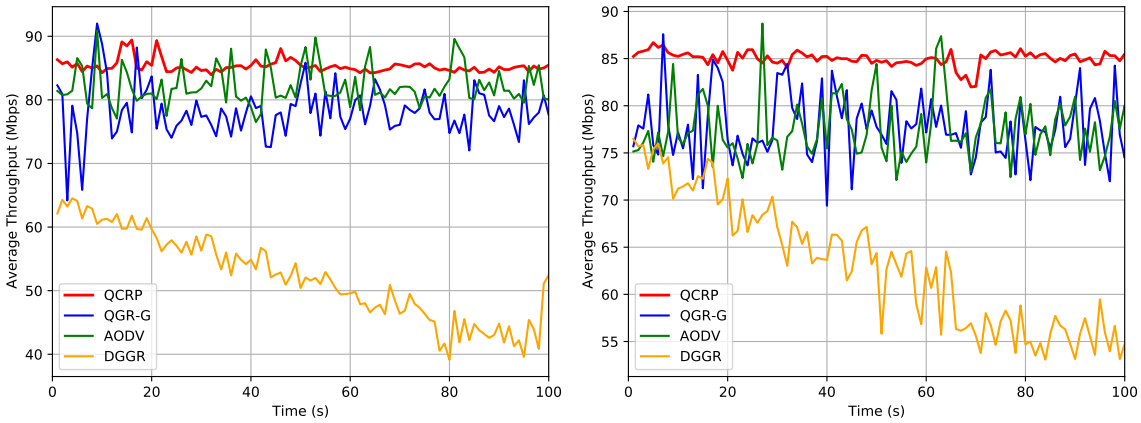
[12]We replaced the carry-and-forward mechanism in DGGR with V2I links for data transmission to reduce E2E delay. Here, AODV essentially finds the minimum-hop path for each cluster.

delay requirements when computing the routing paths. The routing path yield by DGGR fails to satisfy the E2E reliability requirement as it always selects the farthest vehicle within its reach for data transmission leading to low link reliability. However, although QGR-G provides better E2E reliability performance over time (improvement up to 28%), it failed to guarantee the E2E reliability due to topology changes between global topology updates at the BS. Thus, the E2E performance cannot be guaranteed.

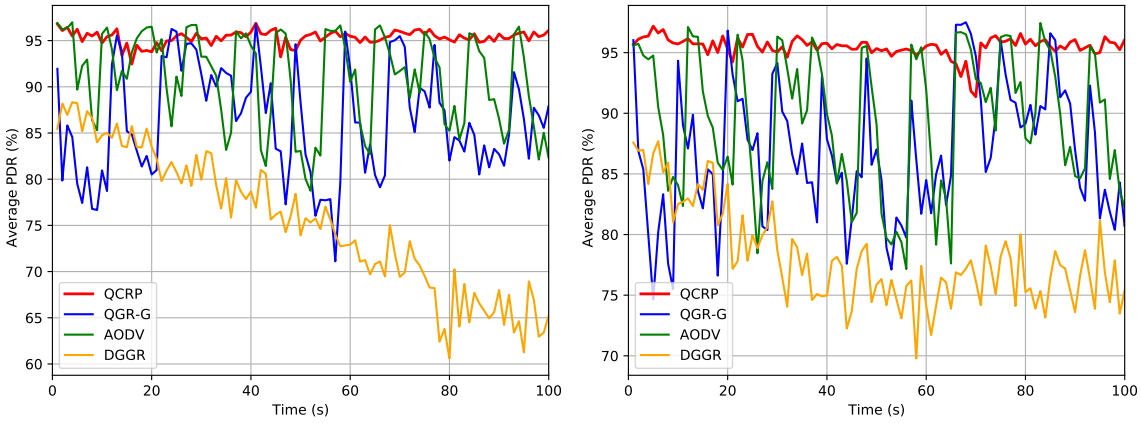
4.6.4 QCRP performance verification

Finally, we evaluate the overall performances of the complete QCRP protocol in a more practical setting using the ns-3 simulator. Using the same channel setting and QoS requirement as summarized in Table 4.3, we implement a constant-bit-rate *on-off* application at the source vehicle sending 10000 500 byte packets at the rate of 90 Mbps to fully utilize the V2V channels. At the beginning of each time period, GPCA computes the global routing path and generates the corresponding forwarding table based on the QGR algorithm. The BS then broadcasts the forwarding table to the network and the packets are relayed accordingly by each relay vehicle. As for in-network control, the forwarding table is updated locally by the LPCAs at the relay vehicles according to the proposed relay selection algorithm. We verify QCRP under different traffic densities in comparison with AODV and DGGR algorithms and adopt QGR-G alone as a benchmark to show the effectiveness of re-routing. In our simulation, we measure the average E2E throughput, latency, and reliability (characterized by PDR) performances, respectively by computing the average E2E QoS performance (i.e., from source to the edge vehicles) of all clusters over one second.

Due to page limit, we only present the simulation results under high density and sparse density for analysis as shown Fig. 4.7. The throughput performance is measured by the number of bits received at the edge vehicle of each cluster over one second and the reliability performance is measured by the number of packets being received by the edge vehicle over the number of packets being sent from the source



(a) Average E2E throughput comparison (High) (b) Average E2E throughput comparison (Sparse)

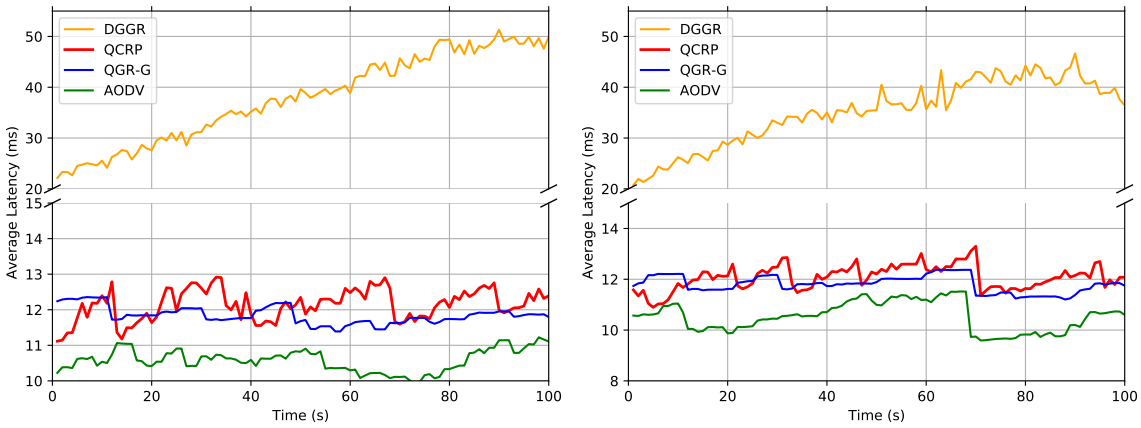


(c) Average E2E PDR comparison (High) (d) Average E2E PDR comparison (Sparse)

Figure 4.7: QCRP E2E performance comparison: first row under high traffic density and second row under sparse traffic density

vehicle. As for the E2E latency performance, we measure the average latency of the packets received by the edge vehicle within one second and compute the average E2E latency of all clusters.

QCRP marked in red curve performed the best over time under various traffic densities thanks to the joint global and local control. Particularly, QCRP showed high robustness to network mobility and traffic densities compared to the comparison methods in terms of both throughput and PDR performances. As shown in Fig. 4.7(a)(b) and Fig. 4.7(c)(d), both AODV and QGR-G marked in green and blue curves, with only global clustering and route planning, suffered from large per-



(e) Average E2E latency comparison (High) (f) Average E2E delay comparison (Sparse)

Figure 4.7: QCRP E2E performance comparison: first row under high traffic density and second row under sparse traffic density

formance variation with oblivious degradation and recovery pattern between each topology update. The variations increased when the traffic density was lower. This is because the V2V links of each cluster are more likely to break due to increased inter-vehicle distances. QCRP showed minimal E2E throughput and reliability performance variation throughout the experiment. Note that the cause of lower PDR in our simulations than in the theoretical analysis is the packet loss due to the limited buffer size we set at each relay vehicle. On the other hand, the E2E throughput performances achieved by DGGR showed short-term variation and long-term decreasing patterns in general due to the lack of global route planning and its greedy-forwarding design. Interestingly, DGGR performed slightly better in throughput when the traffic density was lower. Similar patterns can be observed in its reliability performance. This is because we modified DGGR by replacing its carry-and-forward design with V2I links which are more stable than V2V links.

As it can be observed in Fig. 4.7(d)(f), QCRP showed slightly higher E2E latency with more frequent variations compared to QGR-G and AODV (but always satisfies the E2E latency requirement). This is because a new relay vehicle is re-selected whenever the original relay vehicle is out of the transmission range. Thus, the number

of hops to reach the edge vehicles changes more frequently causing longer latency and higher variations. AODV provisioned the least E2E latency since it uses the least-number-of-hops path for packet transmission. On the other hand, DGGR saw an increasing E2E latency under both traffic densities over time as more vehicles were injected into the network resulting in more hops. Overall, we conclude that QCRP provides improved and robust E2E throughput, reliability, and latency performance under high mobility with the help of joint global and local protocol control.

4.7 Conclusion

In this chapter, we investigate the network routing problem in vehicular networks. We propose a novel network architecture that leverages cross-layer information to generate corresponding control strategies. Following the architecture, a novel routing protocol named QCRP is proposed to guarantee the required QoS. Specifically, a predictive clustering algorithm based on LSTM is proposed to ensure network connectivity by dynamically adjusting the topology update period and network mobility prediction. A QoS-guaranteed routing algorithm is developed to perform route planning in a centralized manner based on QoS requirements and global network topology. In addition, a residual link lifetime-based re-routing algorithm is proposed to adapt to network dynamics and maintain QoS performance. We evaluate our solution with the state-of-art and benchmark using traffic traces with different densities. The results show that our solution is capable of guaranteeing the required QoS performances.

However, the potential of the proposed architecture and QCRP is yet to be explored. For example, how QCRP can be extended to multiple source scenarios and enable serving multiple applications with various QoS requirements can be further investigated.

Chapter 5

QMAC: Resource Allocation for Advanced Vehicular-to-Everything (V2X) Application

In this chapter, we continue to research on how to provide QoS-guaranteed services in vehicular networks. In Chapter 4, we focus on the routing problem in vehicular networks with the assumption of dedicated spectrum bandwidth to support inter-vehicle communications. However, the spectrum is limited and shared among multiple users to improve resource efficiency which results in severe co-channel interference and impedes the user experience. Therefore, in this chapter, we focus on the resource allocation problem in vehicular networks to support stringent QoS requirements and optimize resource efficiency. The frequently used symbols and notations are summarized in Table 5.1.

5.1 Introduction

The resource allocation problem in V2X communications has been heavily explored in the literature [54, 117]. While these approaches show satisfactory performance in dif-

Table 5.1: Notations and definitions

Symbol	Definitions
$u_{i,k}$	k th vehicle located in cluster i
$\mathcal{U},\ \mathcal{M},\ \mathcal{Q}$	Sets of vehicles, clusters, and QoS requirements
$\beta_{i,k}^{(l)}$	Selection indicator of sub-channel l by $u_{i,k}$
$\boldsymbol{\beta}$	Sets of sub-channels
$B_I,\ B_V$	Bandwidth allocated for V2I and V2V links
$P_{BS},\ P_v$	Tx* power used by BS and vehicles for V2I links
$P_{i,k}$	Tx* power used by $u_{i,k}$'s V2V link
$\mathcal{P}$	Sets of power levels
$h_{i,k}$	Channel gain between $u_{i,k}$ and its receiver
$\Gamma_{i,k}^{(l)}$	SINR at $u_{i,k}$ over sub-channel l
$R_{i,k}$	Achievable link capacity of $u_{i,k}$'s V2V link
$\mathrm{Pr}_{i,k}$	Achievable link reliability of $u_{i,k}$'s V2V link
$\phi_{i,k}$	Error control decision indicator of $u_{i,k}$
λ_i	Resource efficiency metric achieved by cluster i
$a_{i,k}$	The action selected by $u_{i,k}$
$g_{i,k}$	The utility gain achieved by $u_{i,k}$ after an action

* Short for transmission

ferent scenarios, many of them only optimize one aspect of QoS metrics, e.g., latency, throughput, and reliability, and assume sufficient spectrum bandwidth and accurate vehicular channel information, which is insufficient for V2X applications requiring guaranteed QoS in multiple dimensions under high mobility. Recently, with the advent of mobile edge computing (MEC) and deep learning (DL) algorithms, researchers propose to adopt reinforcement learning (RL) in V2X scenarios to improve network performance [110]. However, these DL-based solutions suffer from high training costs and are vulnerable to the out-of-distribution problem.

To bridge the gap, we propose a QoS-guaranteed medium access control (QMAC) protocol combining both centralized and distributed control decision leveraging the off-the-shelf tools defined in 5G NR-V2X standards and lightweight online learning

algorithm. Specifically, we first analyze the channel model and reliability model of the target vehicular network. We define a new metric considering both the achievable data rate and spectrum usage to characterize the spectrum resource usage efficiency. Next, we formulate an optimization problem to maximize the proposed resource efficiency metric while considering the QoS requirements as constraints.

Since the formulated problem is NP-hard, we develop a heuristic framework to solve it by decoupling the original problem into two steps. In the first step, the base station (BS) computes the optimal spectrum resource allocation scheme for each vehicle based on the global network knowledge. To reduce the high control overhead caused by frequent topology updates, maintain the network performance, and improve the responsiveness to network dynamics, we enable distributed power control and error control by each relay vehicle in the second step. Before the global network is updated again, the relay vehicles continuously evaluate the current control strategy based on local channel condition observations. A new power level and error control scheme will be selected if the previous ones fail to satisfy the QoS requirement.

Since we assume no collaborations among vehicles, we formulate the distributed power and error control selection as an unknown general-sum game to maximize each relay vehicle's utility and converge to equilibrium over time. A multi-arm-bandit (MAB)-based learning algorithm [60] is adopted to solve the unknown game and yield the optimal control strategies. Finally, we compare our solution with the existing solution and two other baseline approaches via extensive simulation. The results show that our solution is capable of ensuring high resource efficiency while guaranteeing QoS performance.

The rest of the chapter is organized as follows. In Section. 5.2, we introduce the preliminaries and related work. In Section. 5.3, we explain the system model of our scenario. Next, we formulate our resource optimization problem in Section. 5.4. In Section. 5.5, we explain our proposed solution to the optimization problem. Simulation results are discussed in Section. 5.6. Finally, we conclude our work and discuss the future works in Section 5.7.

5.2 Related Work

5.2.1 5G NR-V2X

Two frequency ranges are supported in Rel. 16 NR V2X sidelinks [43], namely frequency range 1 (FR1): 410 MHz −7.125 GHz, and frequency range 2 (FR2): 24.25 GHz −52.6 GHz. In this work, we only consider the resource allocation in FR1, where the cyclic-prefix orthogonal frequency division multiplexing (CP-OFDM) with scalable numerology is adopted for signaling[1]. In NR V2X, a sidelink radio frame is divided into 10 equal-length subframes, each with a duration of 1 ms. The number of slots per subframe depends on the subcarrier spacing (SCS) given by $2^{\mu} \times 15$ kHz used for different numerology, where $\mu \in \{0, 1, 2, 3\}$ is the configuration factor. Thus, the length of a slot is $1/2^{\mu}$ ms.

In NR-V2X, the bandwidth part (BWP) concept is introduced to support devices that cannot handle large bandwidths, enabling flexible and efficient resource usage [43]. Within a BWP, a subset of available resources named resource pool (RP) is configured to be used by users. In the frequency domain, an RP is divided into L contiguous sub-channels denoted by $\mathcal{L} = \{1, 2, \ldots, L\}$ and the bandwidth of each sub-channel is denoted by B_0. In the time domain, the RP is divided into slots and the slot duration is determined by the numerology selected as discussed above. Thus, a sub-channel within a slot can be represented by a resource block group (RBG).

In addition, two modes are defined in 5G NR V2X to facilitate efficient V2X communications. In mode 1, the BS allocates sidelink (SL) resources for V2V communications using the NR uu interface in a centralized manner. Two methods, configured grant (CG) and dynamic grant (DG) are defined in mode 1, where the user entity (UE) requests resources in the CG method while BS reserves resources for UEs in the DG method. In mode 2, UEs can independently select SL resources from the RP and reserve the corresponding sub-channels for a period of time named resource

[1]The selection of numerology depends on the channel condition [20] and is beyond the scope of this chapter.

reservation interval (RRI).

Furthermore, HARQ is supported in NR-V2X to improve reliability performance by employing forward error correction (FEC) and error detection codes in combination with a retransmission strategy. Another new feature introduced in NR-V2X is the re-evaluation mechanism. A UE that has selected one SL resource will continue sensing the transmissions from other UEs during the selection window. A UE can decide whether to transmit by checking if the selected resource is still available. The re-evaluation mechanism adds more flexibility to V2X communications [43].

5.2.2 Resource Allocation Solutions in V2X

The existing solutions can be categorized into centralized and distributed ones in general. In terms of centralized solutions, in [137], a two-stage centralized resource allocation and distributed power control using the non-orthogonal multiple access (NOMA) technology was developed to reduce latency and improve network capacity. A two-step resource allocation strategy was designed in [131] where a joint power control, platoon formation, and resource allocation optimization algorithm based on dynamic programming is developed. Authors in [48] discussed a reliability and latency-aware resource allocation that maximizes the network throughput by power allocation and spectrum reuse. [25] investigated a joint platoon partition, power control and spectrum allocation problem for vehicle platooning to guarantee the reliability of intra-platoon communication and maximize the network capacity. While the centralized approaches are effective for platoons where the network topology is stable, they introduce high control overhead to obtain vehicular channel information in other scenarios where the network topology changes frequently.

As for distributed solutions, most existing works focus on analyzing the collision probability and solving the hidden terminal issue [123]. [118] proposed a receiver grant-based resource allocation strategy to reduce latency and collision probability via explicit resource reservation indication. [69] proposed an enhanced mode 2 procedure using convolutional neural network (CNN) based classifiers to reduce the number of

blind decodings.

More recently, deep reinforcement learning (DRL) has been adopted in vehicular networks to add intelligence for efficient resource usage [89]. In [22], the authors developed an improved random selection (IRS) scheme to reduce the collision probability based on the analysis of packet collision factors. Furthermore, they developed a deep deterministic policy gradient (DDPG) algorithm to improve the network performance via inter-vehicle collaboration. In [47, 77, 107], the authors proposed distributed resource allocation frameworks based on multi-agent reinforcement learning (MARL) to enable vehicles to intelligently re-configure resource allocation policy via interactions with the network environment in a distributed manner. However, due to the lack of global network knowledge and high contention delay, the distributed solutions are insufficient to guarantee the QoS requirements. Moreover, the training costs of DRL models are high and require re-training in new scenarios.

5.3 System Model

5.3.1 Scenario Description

We follow the same network architecture as proposed in Chapter 4, where the network is clustered and routing paths are formed for each cluster using QCRP. As shown in Fig. 5.1, in the proposed framework, each member vehicle uploads its mobility information to the BS periodically via the physical uplink control channel (PUCCH) where a global network topology is maintained. The control decisions on cluster and routing paths are then broadcast to the network via the physical downlink control channel (PDCCH). To reduce the overhead of frequent global topology updates and enable fast response to local network dynamics, the relay vehicles can adjust the original routing path based on local observation by interacting with their receivers via the physical sidelink control channel (PSCCH). The physical sidelink shared channel (PSCCH) is used for data transmission among vehicles.

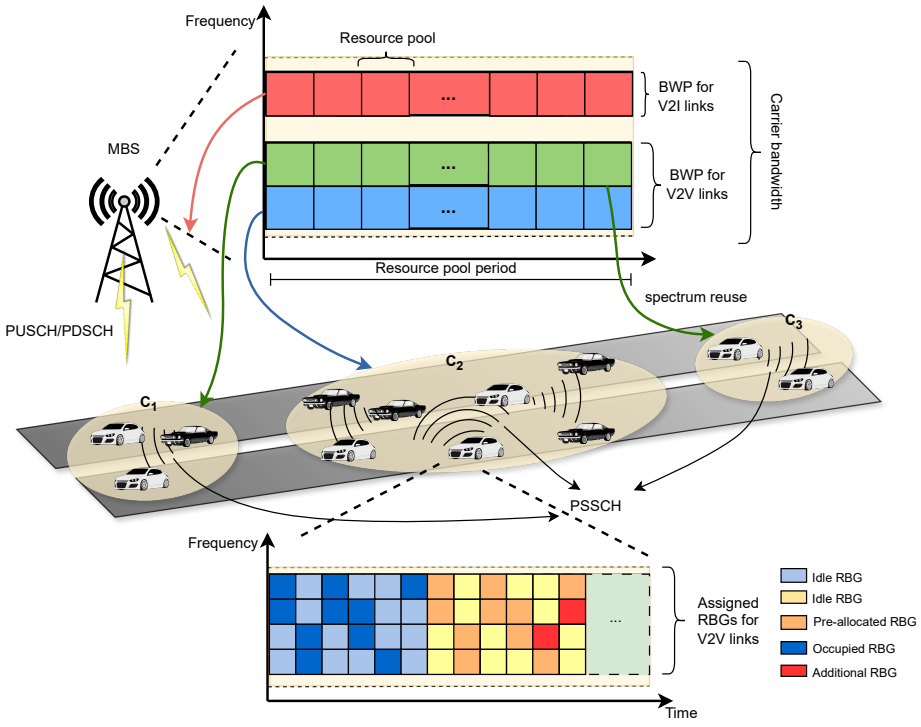


Figure 5.1: An overview of the extended sensor sharing application on a multi-lane highway scenario, where dedicated bandwidth is divided for V2I and V2V communications.

We denote T as the network topology update period at the BS. At the beginning of T, the BS allocates spectrum resources to each cluster to support the required QoS performances. During T, each relay vehicle adjusts the control strategy such as error control and power control independently to adapt to the channel variations.

To be more specific, the BS first divides the total available bandwidth into two parts for V2I and V2V communications, respectively. We denote the total bandwidth dedicated to the V2X application by B_{tot}, and the bandwidth assigned to V2I and V2V communications are denoted by B_I and B_V. B_I and B_V use non-overlapping frequency band for data transmission and thus, the interference between V2I and V2V links is avoided. Next, it allocates B_V to each cluster with the goal of resource efficiency maximization. In this context, we aim to maximize bandwidth spatial reuse by minimizing the number of sub-channels for data transmission while satisfying the

QoS requirements. Within each cluster, we assume the adoption of the time division multiplexing (TDM) so the intra-cluster co-channel interference can be avoided [77].

Furthermore, to address the performance degradation issue caused by network mobility, a distributed power and error control selection algorithm is adopted by each relay vehicle. Each relay vehicle continuously evaluates its channel condition via the CSI feedback sent from its receivers and adjusts the transmission power and error control scheme if necessary.

5.3.2 Network Model

Denote $\mathcal{U}$ as the complete set of member vehicles. In total, there are U vehicles. Each vehicle uploads its mobility information denoted by $\{l_u, v_u\},\ u \in \mathcal{U}$ to the BS, where $l_u = (x_u, y_u)$ is the location of vehicle u and v_u is its speed. Thus, the inter-vehicle distance can be computed:

$$d_{u,u'} = \sqrt{(x_u - x_{u'})^2 + (y_u - y_u)^2},\ u, u' \in \mathcal{U}. \tag{5.1}$$

We assume the network is divided into M clusters, denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$. Within a cluster i, we denote the set of relay vehicles by $\mathcal{K}_i$ and the k th relay vehicle in cluster i by $u_{i,k}$. In addition, we denote the QoS requirements by a tuple $\mathcal{Q} = \{\theta_{\text{rel}}, \theta_{\text{rate}}, \theta_{\text{SINR}}\}$, where θ_{rel} is the reliability requirement, θ_{rate} is the data rate requirement, and θ_{SINR} is the SINR requirement. In this work, we assume $\mathcal{Q}$ is known by all vehicles and remains unchanged during the same session.

5.3.3 Channel Model

We assume line-of-sight for V2I communications between BS and each relay vehicle. Denote the transmission power of BS by P_{BS}, then the signal-to-noise ratio (SNR) of a downlink channel can be expressed as

$$SNR_d = \frac{P_{BS}}{N_0} \delta d_{BS,Rx}^{-\alpha} |h|^2, \tag{5.2}$$

where $d_{BS,Rx}$ is the distance between the BS to the receiver vehicle, δ is the shadowing component, h is the Rayleigh-distributed fading magnitude with $\mathbb{E}(|h|^2) = 1$, N_0 is the power of additive white Gaussian noise (AWGN), and α is the path loss component. We assume a symmetric channel condition for the uplink. Thus, the SNR of the uplink can be expressed as

$$SNR_u = \frac{P_v}{N_0} \delta d_{Tx,BS}^{-\alpha} |h|^2, \tag{5.3}$$

where $d_{Tx,BS}$ is the distance between the source vehicle and BS and P_v is the vehicle transmission power.

Using the well-known Shannon capacity model, the uplink and downlink data rate is given by[2]:

$$R_f = B_I \cdot \log_2 (1 + SNR_f), \ f \in \{u, d\}. \tag{5.4}$$

As for the V2V links, since the same sub-channels are shared by multiple vehicles in different clusters, we adopt SINR to model the transmission rate. Here, we denote $\Gamma_{i,k}^{(l)}$ as the SINR experienced by $u_{i,k}$ at sub-channel l, whose received signal is interfered by other vehicles $u_{i',k'}$, $i' \neq i, i' \in \mathcal{M}$, $k' \in \mathcal{K}_{i'}$ in different clusters that reuses the same sub-channel:

$$\Gamma_{i,k}^{(l)} = \frac{P_{i,k} |h_{i,k}^{(l)}|^2}{\sigma_0^2 + \sum_{i'=1, i' \neq i}^{M} \sum_{k' \in \mathcal{K}_{i'}} \sum_{l=1}^{L} \beta_{i,k}^{(l)} P_{i',k'} |h_{i',k'}^{(l)}|^2}, \tag{5.5}$$

where $P_{i,k}$ is the vehicle transmission power; $|h_{i,k}^{(l)}|^2 = \delta d_{i,k}^{-\alpha} |h|^2$ is the channel gain from the sender to $u_{i,k}$, where $|h|^2 \sim \mathcal{N}(0, 1)$ is the Gaussian variable representing the Rayleigh fading; σ_0 is the variance of the AWGN; Similarly, $P_{i',k'}$ and $|h_{i',k'}^{(l)}|$ is the transmission power and channel gain from the interfering vehicle $u_{i',k'}$ to $u_{i,k}$. Let $\beta_{i,k}^{(l)} \in \{0, 1\}$ denote allocating sub-channel l to $u_{i,k}$. Thus, the data rate for the

[2]We assume the BS multicast the control decision to the network using the entire B_I, so co-channel interference is omitted.

corresponding V2V link can be modeled as follows:

$$R_{i,k} = \sum_{l=1}^{L} \beta_{i,k}^{(l)} B_0 \log_2(1 + \Gamma_{i,k}^{(l)}),\ i \in \mathcal{M}, k \in \mathcal{K}_i. \tag{5.6}$$

5.3.4 Reliability Model

We use the packet delivery ratio (PDR) of a link to characterize the per-hop transmission reliability:

$$P_d = (1 - p_b)^n, \tag{5.7}$$

where p_b is the bit error rate (BER) and n is the payload in bits. PDR can be improved using physical-layer error correction code (ECC) such as forward error correction (FEC) by adding redundant information bits [86, 136]. Let $w_{i,k}$ denote the reliability improvement gain expressed as

$$w_{i,k} = \begin{cases} \epsilon, & \text{if FEC is adopted,} \\ 1, & \text{otherwise,} \end{cases} \tag{5.8}$$

where ϵ is the improvement gain added to PDR when FEC is adopted. For example, m bits are used to encode the original data packet and successfully decoding n ($m > n$) bits at the receiver side will result in successful packet delivery. In this context, the improvement gain of adopting FEC can be expressed as

$$\epsilon = \tbinom{m}{n} \frac{(1 - p_b)^n p_b^{m-n}}{(1 - p_b)^n} = \tbinom{m}{n} p_b^{m-n}. \tag{5.9}$$

Moreover, since the HARQ mechanism is introduced in NR V2X SL to provide reliable transmission for sidelink communications, the reliability can be further improved by reserving an additional slot for packet retransmission as shown in Fig. 5.2. Let $\eta_{i,k}$ denotes the improvement on reliability using retransmission at $u_{i,k}$, then it

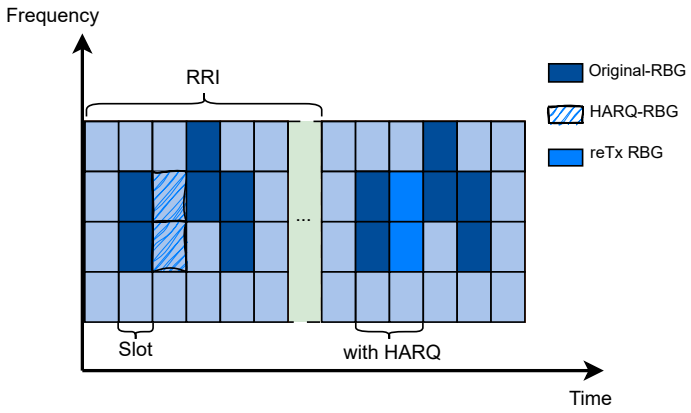


Figure 5.2: An example for periodic resource reservation considering HARQ retransmission.

can be expressed as:

$$\eta_{i,k} = \begin{cases} \frac{1-(1-P_d)^2}{P_d}, & \text{one-time retransmission,} \\ 1, & \text{otherwise.} \end{cases} \tag{5.10}$$

In this context, the achievable PDR of a V2V link for $u_{i,k}$ can be written as follows

$$Pr_{i,k} = \eta_{i,k} \cdot w_{i,k} \cdot P_d. \tag{5.11}$$

We use $\phi_{i,k} = \langle \eta_{i,k}, w_{i,k} \rangle$ as a decision indicator to denote the error control combination selected by $u_{i,k}$, e.g., $\langle 0, 0 \rangle$ represents no error control is needed and $\langle 1, 1 \rangle$ represents both FEC and retransmission is adopted.

5.4 Problem Formulation

5.4.1 Resource Efficiency Optimization

Different from the existing works [131] which optimize the network utility by maximizing the data rate performance, we propose to maximize the resource efficiency in our problem formulation. In other words, we aim at achieving a finer granularity

of resource utilization by avoiding resource over-provision. We assume that the improvement gain on QoS performances becomes trivial after the required resources are satisfied, for example, for an application with a data rate requirement of 80 Mbps, providing a large link capacity at a few hundred Mbps brings limited additional value to the application compared with providing just enough link capacity around 80 Mbps. In this context, we define λ_i, the achievable data rate per resource to characterize the resource efficiency of cluster i within the resource allocation period, i.e.,

$$\lambda_i = \sum_{k \in \mathcal{K}_i} \frac{R_{i,k} \cdot L}{\theta_{\text{rate}} \cdot \sum_{l=1}^{L} \beta_{i,k}^{(l)}}, \tag{5.12}$$

where we use the data rate gain R/θ_{rate} and resource usage ratio $\sum_{l=1}^{L} \beta_{i,k}^{(l)}/L$ to represent the network utilization performance. Thus, we formulate our resource efficiency maximization problem as follows:

$$\mathbf{P_0}: \quad \max_{\boldsymbol{\beta}, \boldsymbol{\Phi}, \boldsymbol{P}} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{M} \lambda_i[t] \tag{5.13a}$$

$$\text{s. t.} \quad \beta_{i,k}^{(l)} \in \{0,1\}, \ l \in \mathcal{L}, k \in \mathcal{K}_i, i \in \mathcal{M}, \tag{5.13b}$$

$$B_I + B_V \leq B_{tot}, \tag{5.13c}$$

$$R_{i,k}[t] \geq \theta_{\text{rate}}, \ i \in \mathcal{M}, k \in \mathcal{K}_i, t = 1, \ldots, T, \tag{5.13d}$$

$$R_f[t] \geq \theta_{\text{rate}}, \ \forall f \in \{u, d\}, t = 1, \ldots, T, \tag{5.13e}$$

$$Pr_{i,k}[t] \geq \theta_{\text{rel}}, \ k \in \mathcal{K}_i, i \in \mathcal{M}, t = 1, \ldots, T, \tag{5.13f}$$

where $\boldsymbol{\beta}, \boldsymbol{\Phi}, \boldsymbol{P}$ are the control decision vectors of sub-channel allocation, error control scheme combination, and transmission power selection determined for all relay vehicles, respectively; (5.13b) is the resource selection constraint, i.e., $\beta_{i,k}^{(l)} = 1$ if the l th sub-channel is allocated to vehicle $u_{i,k}$ and $\beta_{i,k}^{(l)} = 0$, otherwise; (5.13c) indicates the bandwidth usage constraint; (5.13d)-(5.13f) represent the data rate, transmission period, and reliability constraints, respectively. Note that we assume B_I is assigned with the number of sub-channels to support the throughput requirement. Thus, it

can be determined by solving (5.13e) when the equation holds. B_V can be calculated by counting the total number of reserved sub-channels, i.e.,

$$B_V = \max \left\{ \sum_{l=1}^{L} \beta_{i,k}^{(l)} B_0 | \forall k \in \mathcal{K}_i, i \in \mathcal{M} \right\}. \tag{5.14}$$

The main challenge of solving this optimization problem lies in estimating the achievable data rate and the reliability performances given the time-varying channel condition due to high network mobility, inter-cluster co-channel interference, and fading effect.

5.4.2 Power and Error Control Selection Game

We consider a set of discrete transmission power levels $\mathcal{P} := \{P_1, \ldots, P_C\}$ with size C and an error control set of size W denoted by $\Phi := \{\phi_1, \ldots, \phi_W\}$. The selection of power level and error control combinations together form a finite action set $\mathcal{A} := \{a_j = (P_j, \phi_j) | j = 1, \ldots, A\}$, where $A = C \times W$ is the total number of actions. Assume that a vehicle can finish the interaction in one round of the game[3], i.e., select an action (power level and error control scheme) for packet transmission and receive the corresponding ACK. At the end of each round t, the relay vehicle can calculate the utility of the action $g^t(a_n)$. To highlight the dependency of the utility gained by vehicle $u_{i,k}$ in round t on all vehicles sharing the same sub-channel, we denote the utility function by $g_{i,k}^t(a_n^t; A_{-n}^t)$, where A_{-n}^t denote the actions taken by other relay vehicles.

Note that each relay vehicle is in the bandit setting, i.e., only its own utility can be calculated by measuring the throughput and any other vehicles' actions and utilities can not be observed. The goal of each relay vehicle in the unknown general-sum game is to maximize its own utility while not hurting the overall network performance. We use the correlated equilibrium concept to measure the optimality of a solution.

[3]We assume that the duration of each round is the same as the resource reservation interval pre-configured for the relay vehicles.

Definition Let $\mathbf{P}^t$ be the joint probability distribution over the complete action space $\mathbb{A} := \prod_{u \in \mathcal{U}} A_u$ among all relay vehicles sharing the same spectrum band in round t, where t is the time slot length of each subframe configured by the BS at during the centralized configuration phase. We define $\mathbf{P}^t$ as the correlated equilibrium if for every vehicle, the incentive to deviate from one action a_n to another a'_n satisfies:

$$\sum_{a \in \mathbb{A}} \mathbf{P}^t(g(a'_n; A_{-n}) \leq \sum_{a \in \mathbb{A}} \mathbf{P}^t g(a_n; A_{-n})), \tag{5.15}$$

i.e., no vehicle can improve its utility via strategy modification.

We adopt the swap regret concept to characterize the action selection process of each vehicle in the proposed general-sum game. We define the swap function ϕ that maps action a_n to a'_n, $a_n, a'_n \in A$. Then the swap regret during the T-round game can be expressed as:

$$R_{swp} = \max \mathbb{E} \left[\sum_{t=1}^{T} \sum_{a_n \in A} g^t_{i,k}(\phi(a^t_n); A^t_{-n}) - g^t_{i,k}(a^t_n; A^t_{-n}) \right]. \tag{5.16}$$

Therefore, we aim to design an algorithm to find the optimal control decision for each relay vehicle by minimizing the swap regret.

5.5 QoS-guaranteed Medium Access Control

We aim to guarantee the QoS requirements using a hybrid control strategy: centralized resource allocation and distributed resource adaptation. Since $\mathbf{P_0}$ is a mixed-integer nonlinear programming (MINLP) which is NP-hard, we develop a heuristic solution by decoupling it into two subproblems and solve them separately at different places as depicted in Fig. 5.3.

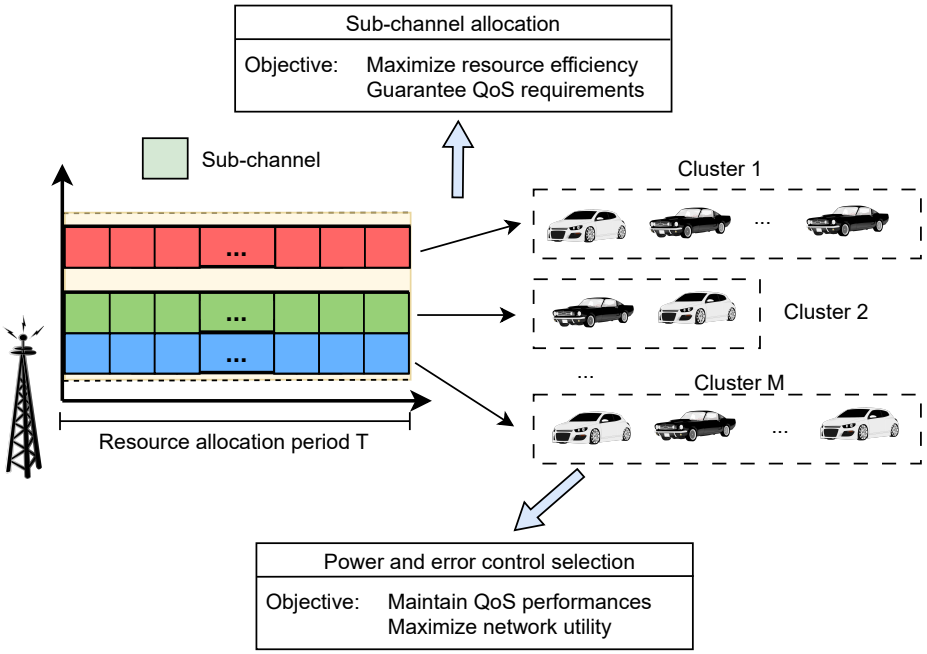


Figure 5.3: Overview of the proposed hybrid control strategy: centralized control at BS and distributed control at relay vehicles.

5.5.1 Branch and Bound-based Centralized Resource Allocation at BS

The BS is responsible for computing the optimal subchannel allocation scheme based on the global network topology knowledge and QoS requirements in a centralized fashion. We first decouple $\mathbf{P_0}$ by abstracting the sub-channel selection problem.

$$\mathbf{P_1}: \min_{\boldsymbol{\beta}} \sum_{i=1}^{M} \sum_{l=1}^{B_V} \beta_i^{(l)} \tag{5.17a}$$

$$\text{s. t. } \beta_i^{(l)} \in \{0,1\},\ l \in \mathcal{L}, i \in \mathcal{M}, \tag{5.17b}$$

$$B_I + B_V \leq B_{tot}, \tag{5.17c}$$

$$R_i \geq \theta_{\text{rate}},\ i \in \mathcal{M}, \tag{5.17d}$$

$$Pr_i \geq \theta_{\text{rel}},\ i \in \mathcal{M}. \tag{5.17e}$$

We develop a two-step algorithm using the branch and bound (BB) method to solve $\mathbf{P_1}$ as summarized in Algorithm 10. First, we compute the minimum bandwidth required for V2I communications (line: 3) by

$$B_I = \frac{\theta_{\text{rate}}}{\log(1 + SNR_{d,\min})}, \tag{5.18}$$

where $SNR_{d,\min}$ is the minimum SNR measured among all cluster heads. Then, the rest of the sub-channels are reserved for V2V communications, i.e.,

$$B_V = B_{tot} - B_I. \tag{5.19}$$

Next, we compute the optimal resource allocation strategy for each cluster (line: 4). Specifically, we start with relaxing the binary sub-channel selection variable $\beta_i^{(l)}$ to a continuous variable $\hat{\beta}_i^{(l)} \in [0, 1]$ and thus, $\mathbf{P_1}$ is converted into a convex nonlinear programming denoted by $\mathbf{P_1'}$ and can be solved using a standard optimization solver. After solving $\mathbf{P_1'}$, we map the continuous variable $\hat{\boldsymbol{\beta}}$ back to discrete values while meeting the QoS constraints if there exists $\hat{\beta}_i^{(l)} \notin \{0, 1\}$ (line: $5 - 17$). To speed up the convergence of the algorithm, we scale each $\hat{\beta}_i^{(l)}$ by a threshold value ϱ,

$$\hat{\beta}_i^{(l)} = \begin{cases} 1, & \text{if } \hat{\beta}_i^{(l)} \geq \varrho, \\ 0, & \text{otherwise.} \end{cases} \tag{5.20}$$

Then, the newly scaled decision set is evaluated by the QoS constraints iteratively (line: $8 - 16$). If all the constraints are met, we continue to minimize bandwidth usage by flipping the sub-channel $\beta_i^{(l)}$ with the minimum $SINR$ from 1 to 0 of the cluster achieving the highest data rate, i.e.,

$$\{\beta_{i^*}^{(l)} := 1 \rightarrow 0 | l^* = \arg\min_{l \in \mathcal{L}} SINR_{i^*}^{(l)}, i^* = \arg\max_{i \in \mathcal{M}} R_i, \}. \tag{5.21}$$

Otherwise, an error message will be pushed to the BS to warn of the lack of spectrum

Algorithm 10 Branch and Bound based Spectrum Allocation

Input: Network topology, QoS requirements $\mathcal{Q}$.
Output: Sub-channel allocation strategy $\boldsymbol{\beta}$
Compute available bandwidth resources: solve (5.18) and (5.19) for B_I and B_V.
Relax (5.17b) to $[0, 1]$ and obtain $\mathbf{P}'_\mathbf{1}$
Selection decision initialization: Solve $\mathbf{P}'_\mathbf{1}$ for $\hat{\boldsymbol{\beta}}$.
if $\forall \hat{\beta}_i^{(l)} \in \{0, 1\}$, $\hat{\beta}_i^{(l)} \in \hat{\boldsymbol{\beta}}$. **then**
    Return $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$.
else
    Scalarize $\hat{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$ by probability using (5.20).
    Evaluate $\boldsymbol{\beta}$ by constraints (5.17b)-(5.17e).
    Push error message to the application if exists failed constraint.
    while all constraints are met **do**
        Flip $\beta_i^{(l)}$ following (5.21).
        if exists failed constraint **then**
            Return unfliped $\boldsymbol{\beta}$.
        end if
    end while
end if

resources to guarantee the required QoS. The updated decision set is then evaluated again following the same procedure until a feasible solution is found.

5.5.2 Control Strategy Adjustment with Bandit Feedback

Before the global topology is updated, i.e., during the distributed control period, each relay vehicle adjusts its transmission power and error control scheme by minimizing the swap regret defined in (5.16).

We assume the adoption of the re-evaluation mechanism introduced in the NR V2X Sidelink standard [43] in our scenario, where all vehicles can sense and manage resources independently. As shown in Fig. 5.4, each relay vehicle maintains a sensing window for strategy evaluation and a selection window for strategy selection[4]. As it has been discussed in Section 5.4.2, we denote $\mathcal{A}$ as the complete combinations of all error control schemes and power levels and $a_{i,k}[t] = \{\phi_{i,k}, P_{i,k}\}$ as the control

[4]Thanks to the centralized allocation, each relay vehicle is allocated a set of specific sub-channels for data transmission, and thus, the hidden terminal problem is avoided.

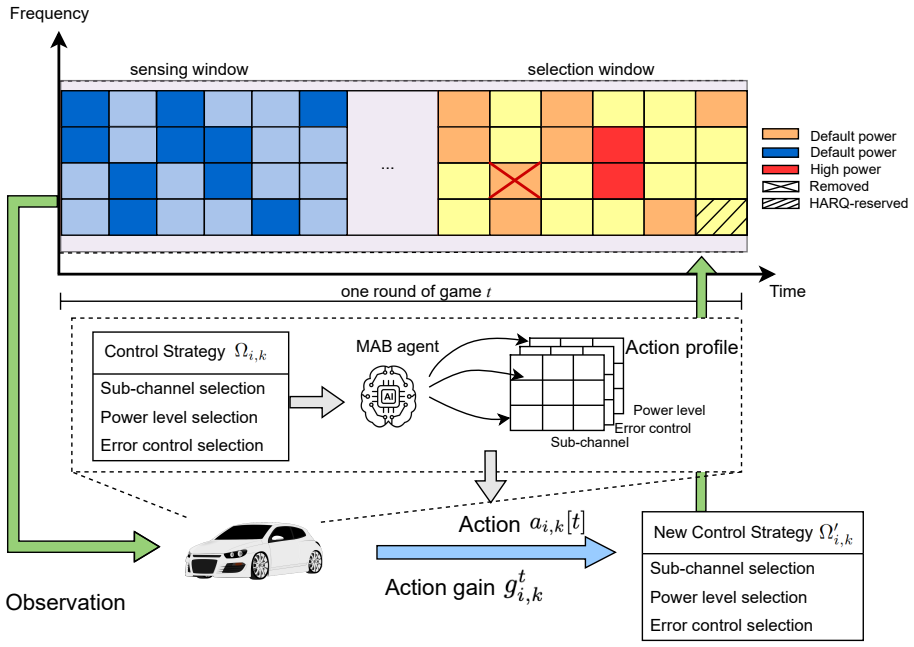


Figure 5.4: MAB-based control strategy adjustment. The relay vehicle evaluates the previous control strategy during the sensing window and configures the new strategy in the selection window.

strategy decided by each relay vehicle $u_{i,k}$ $\forall i \in \mathcal{M}, k \in \mathcal{K}_i$ at the beginning of t th transmission period. During the sensing window, $u_{i,k}$ continuously senses the SINR of the assigned sub-channels $\boldsymbol{\beta}_{i,k}$ via the CSI feedback from its corresponding receivers and computes the achieved utility gain $g^t_{i,k}$. We design the utility function as the achievable goodput:

$$g^t_{i,k}(a^t_n; A^t_n) = Pr_{i,k} \cdot C(R_{i,k}), \tag{5.22}$$

where $Pr_{i,k}$ is the probability of successful packet transmission and $C(R_{i,k})$ is a clip function applied to the achieved data rate: $C(R_{i,k}) = 1$ when $R_{i,k} \geq R_{\text{rate}}$ and $C(R_{i,k}) = R_{i,k}/R_{\text{rate}}$, otherwise. In this case, the utility is limited within the range of $[0, 1]$. The utility function can be easily extended to incorporate other factors such as delay, spectrum, and power efficiency, etc. In this work, we focus on the goodput performance.

Next, each relay vehicle $u_{i,k}$ adjusts the current control strategy in the selection window based on the CSI feedback. The change of the control strategy is determined

by solving the unknown general-sum game we formulated in Section. 5.4.2 where the swap regret defined in (5.16) is minimized.

To this end, we design a learning framework using the MAB concept as shown in Fig. 5.4. Each relay vehicle needs to determine a set of control strategies to maintain the QoS performance based on the CSI observations. This process can be regarded as an agent (i.e., the vehicle) playing a non-stochastic multi-armed bandit (i.e., the control strategy set) against a non-oblivious adversary (i.e., the competing vehicles) [12]. In this context, the vehicle needs to tradeoff the exploration (i.e., understand the payoff of choosing other actions) and exploitation (i.e., continuously choosing the action that seems to be optimal) to maximize its utility.

Since not all actions (i.e., power level and error control selection) are sufficient to guarantee the QoS performances and exploring the complete action space $\mathcal{A}$ can be time-consuming, we first filter out the inapplicable actions by the data rate and reliability constraints:

$$\begin{cases} \sum_{l\in\mathcal{L}_i} \beta^{(l)} B_0 \log(1+\frac{P_t'}{P_{t-1}}\cdot\Gamma_{i,k}) \geq \theta_{\text{rate}}, \\ \phi_{i,k}'\cdot P_d \geq \theta_{\text{rate}}, \end{cases} \tag{5.23}$$

where P_{t-1} is the power level selected in the previous round $t-1$ and P_t' represents any power level that satisfies the data rate constraint in the coming round t based on the SINR observations in the sensing window. Similarly, $\forall\phi_{i,k}'\in\Phi$ is considered as an applicable error control scheme. In this context, we denote $\mathcal{A}'=\{\mathbf{P}',\mathbf{\Phi}'\}$ as the remaining action set after filtering. Next, we adopt a swap-regret minimization approach [60] to determine the new control strategy.

At the beginning of each transmission period, i.e., each round of the game, the vehicle selects an action from the filtered action set $\mathcal{A}'$ based on the evaluation of current channel states to maintain link performance. An action is required only when if the current action no longer meets the QoS requirements, otherwise, the control strategy remains unchanged and no action will be taken. The transition of

Algorithm 11 Distributed Control Strategy Adaption

Input: Predefined control strategy set $\mathcal{A}$, QoS requirements $\mathcal{Q}$
Output: Updated control strategy $\mathbf{\Omega}$.
for $t = 1, 2, \ldots, T$ **do**
  for Each relay vehicle $u_{i,j} \in \mathcal{U}$ **do**
    // *Action space filtering*:
    Obtain the applicable action set $\mathcal{A}'$ using (5.23) filter.
    // *Initialize*:
    $q^0_{a,a'} = \frac{1}{A},\ L^0_{a,a'} = 0,\ \forall a, a' \in \mathcal{A}'$.
    // *Procedure*:
    Compute the action selection probability distribution Π^t_n using (5.24).
    Choose an action with probability $a \sim \Pi^t_n$.
    Receive feedback and compute utility gain g^t_a.
    for $a \in \mathcal{A}$ **do**
      Compute the instant penalty $Y^t_{a,a'}$ using (5.25).
      Compute the accumulated penalty $L^t_{a,a'}$ using (5.26).
      Update the transition probability matrix $\mathbf{Q}^t$ using (5.27).
    end for
  end for
end for

actions between each round can be modeled by a Markov chain. The transition matrix denoted by $\mathbf{Q}^t$ is updated at the end of each round based on the obtained utility. Each row of $\mathbf{Q}^t_n$ corresponds to an action transition probability distribution denoted by $Q^t_n := [q^t_{a,a'}]_{a,a' \in \mathcal{A}}$, where a' is the new action and $\sum_{a' \in \mathcal{A}} q^t_{a,a'} = 1$. Similarly, the action selection probability distribution is denoted by $\Pi^t_n := [\pi^t_a]_{a \in \mathcal{A}'}$ and $\sum_{a \in \mathcal{A}} \pi^t_a = 1$, where π^t_a represents the probability of choosing action a in round t. Then, the final decision can be obtained by solving the following equation:

$$\Pi^t_n = \Pi^t_n \cdot \mathbf{Q}^t_n. \tag{5.24}$$

In addition, we define the instant penalty of choosing a new action a' instead of a as follows:

$$Y^t_{a,a'} = \frac{1[a^t_n = a']p^t_a}{p^t_{a'}}(1 - g^t_a), \tag{5.25}$$

where g_a is the utility gain of action a calculated using (5.22). Let $L^{t+1}_{a,a'}$ denote the

variable to track the accumulated penalty during the T rounds of game, where

$$L_{a,a'}^{t+1} = L_{a,a'}^{t} + Y_{a,a'}^{t},\ t = 0, 1, \ldots, T-1. \tag{5.26}$$

The transition probability can be further expressed as:

$$q_{a,a'}^{t+1} = \frac{\exp(-\varphi^t L_{a,a'}^{t})}{\sum_{a' \in \mathcal{A}} \exp(-\varphi^t L_{a,a'}^{t})}, \tag{5.27}$$

where φ^t is a positive non-increasing control variable to determine the learning rate. We use $\varphi^t = \sqrt{\frac{\log A_n}{t}}$ as suggested in [60]. Note that $L_{a,a'}^{t}$ will not increase if an action is not selected as defined in (5.25). Thus, the design of (5.27) has a better ability to trade-off between exploration and exploitation, that is, the action will have a higher probability of being selected if it has not been selected or suffers a lower penalty.

Finally, we summarize the proposed control strategy selection algorithm in Algorithm 11. As shown from Line 3 to Line 6, at the beginning of each control strategy update period, i.e., $t = 0$, the relay vehicle filters out the undesirable actions from the action set, initializes the transition matrix using a uniform distribution $Q_{n,a}^{0} \sim U(0, A)$, and sets the accumulated penalty $L_{a,a'}^{0} = 0$. During the control strategy adaptation period, i.e., $t = 1, \ldots, T$, each relay vehicle adjusts its control strategy by choosing an action from the learned probability matrix P_n^t as shown from Line 9 to Line 11. Then the transition matrix is updated as shown from Line 12 to Line 16. At the end of each round, the action set $\mathcal{A}$ is updated again using the newly received local information as shown in Line 17. We use $\mathbf{\Omega} = \{a_{1,1}[1], \cdots, a_{M,K_M}[1], a_{1,1}[2], \cdots, a_{M,K_M}[T]\}$ to represent the action decision set determined by each relay vehicle during T time slots.

5.5.3 Complexity Analysis

Since QMAC is executed at two different places, i.e., the centralized spectrum allocation at BS (Algorithm 10) and distributed control strategy update at each relay

vehicle (Algorithm 11), we discuss their complexities separately.

For Algorithm 10, we adopt the `fmincon` solver implemented in MATLAB [80] using the sequential quadratic programming (SQP) method to solve the nonlinear convex optimization problem formulated in $\mathbf{P}_1'$. SQP yields a typical quadratic time complexity depending on the input size and constraint complexity [36,103]. Assuming the channel selection space $\boldsymbol{\beta}$ has an average size N, then the time complexity of this step becomes $O(N^2)$. Next, each element in $\boldsymbol{\beta}$ needs to be mapped back to a discrete value by comparing the estimated performance metric with the required QoS values. In the end, the time complexity of Algorithm 10 is $O(N^3)$. However, since Algorithm 10 is running at the BS and can be equipped with MEC server, the actual computing time can be reduced.

Algorithm 11 requires computing the transition probability distribution matrix $\boldsymbol{Q}_n^t$ for each vehicle at time slot t. Each action a_n^t requires $O(A)$ time to update its transition distribution. Thus, the time complexity is $O(A^2)$. In other words, the complexity of Algorithm 11 grows quadratically with the size of the action space $\mathcal{A}$. However, having a finer granularity of transmission power levels or extending error control functions allows better adaptivity to the dynamic environment of the system. Therefore, there exists a tradeoff between complexity and control performance.

5.6 Performance Evaluation

In this section, we verify QMAC via extensive simulation experiments. We first introduce the experiment settings in our simulations, including traffic trace generation, channel parameters, QoS requirements, and benchmarks. Next, we discuss the simulation results in detail to provide a comprehensive analysis of our proposed solution.

Table 5.2: Parameters Setting

Parameters	Values
Fading factor α	3.2
sub-channel bandwidth B_0	300 kHz
Noise power spectrum density	-100 dBm/Hz
Total assigned bandwidth B_{tot}	30 MHz
Maximum V2V Tx power	30 dBm
Default V2V Tx power	23 dBm
Data rate requirement θ_{rate}	80 Mbps
Reliability requirement θ_{rel}	99%
SINR threshold for data decoding θ_{SINR}	10 dB

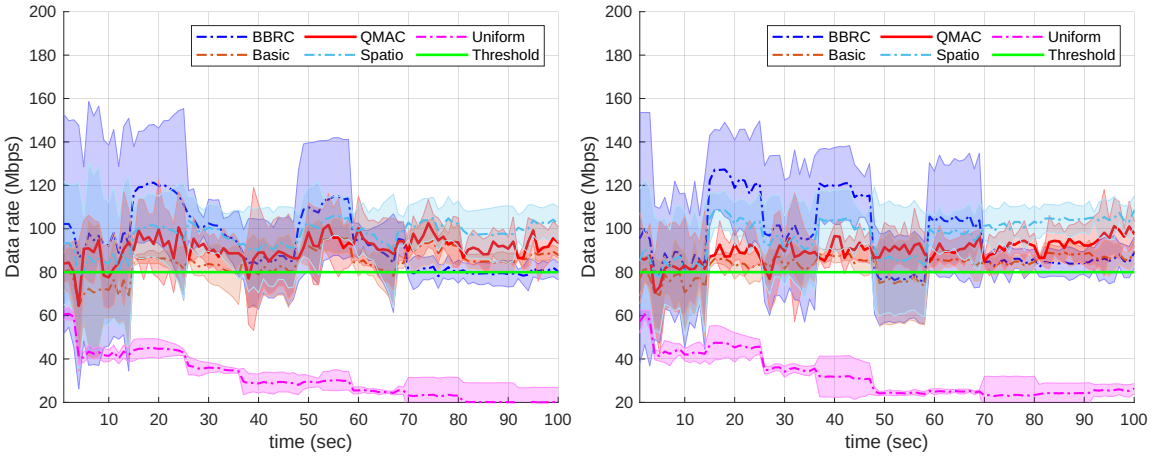
5.6.1 Experiment setting

We used the SUMO simulator to generate four mobility traces with different traffic densities (high, medium, low, and sparse) on a four-lane 5-km highway, where $463, 336, 219$, and 136 vehicles are injected into the system. A BS is deployed in the middle of the road covering the target region. Three types of vehicles with different speed ranges: $(25 - 28$ m/s), $(28 - 35$ m/s), and $(35 - 40$ m/s), are considered. All vehicles adopt the default Krauss car-following model and lane-change model [121] to simulate real car movements.

The vehicular network is first clustered using the QCRP protocol [111] which determines the cluster heads and updates the global network topology and re-clusters every 10 seconds. Then, we apply the proposed solution based on the clustering results[5]. Finally, the physical layer resource parameters and QoS requirements are summarized in Table 5.2.

We compare QMAC with a BB-based solution developed in [131] (denoted by *BBRC*) and two baseline approaches, namely, uniform resource allocation (denoted by *Uniform*) and spatial resource reuse (denoted by *Spatio*). In Uniform, B_V is distributed equally to each cluster, i.e., $B_V^{(i)} = (B_{tot} - B_I)/M,\ i \in \mathcal{M}$. In Spatio, we first

[5]In the simulation, we assume that at each time slot, only one vehicle is transmitting within each cluster. Thus, the transmission process in one cluster is only affected by transmitters from other clusters and the number of interfering vehicles equals to $M - 1$.



(a) Data rate performance under high traffic density

(b) Data rate performance under low traffic density

Figure 5.5: Data rate performance comparison under different traffic densities.

compute the minimum bandwidth needed for supporting the data rate requirement $B_{V,\min}$ using (5.6) and the required SINR value. Then, each cluster is assigned with $B_{V,\min}/B_0$ number of sub-channels and two non-adjacent clusters reuse the same set of sub-channels to reduce interference. In addition, to highlight the effectiveness of the distributed control mechanism empowered by MAB, we add another comparison set denoted by *Basic*, with centralized spectrum allocation only without distributed control at relay vehicles. In our experiments, we compare the data rate, reliability, resource usage ratio, and resource efficiency performances of methods under different traffic densities.

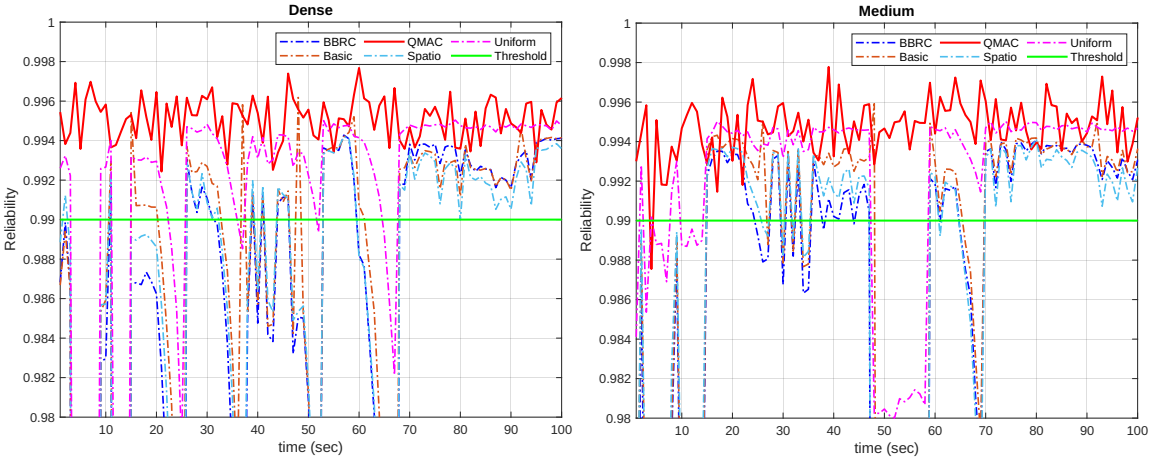
5.6.2 Data rate performance analysis

First, we analyze the data rate performances as shown in Fig. 5.5. We show the average data rate performances (plotted in curves and the shaded areas are their standard deviation) of different schemes under various traffic densities. Here, we only present the simulation results under high traffic density and low traffic density due to page limitation. QMAC finds the optimal control strategy by maximizing the resource efficiency metric while satisfying the QoS requirements. As shown in the red solid line,

the data rate performance is maintained close to the required values, i.e., 80 Mbps (indicated by the solid green line), throughout the experiment. It showed higher variance compared to other approaches due to the existence of distributed power control. Note that QMAC did not achieve the highest data rate performance because it aims at reaching the required value with minimum spectrum usage as explained in Algorithm 10. Basic showed lower data rate performance over time compared to QMAC as plotted in the orange dash-dotted curve due to the lack of efficient power control. On the other hand, BBCR generated control strategies by maximizing the total data rate. Thus, it achieves the highest data rate performance as shown in the blue dash-dotted curves in Fig. 5.5. The pattern is more obvious under the low-traffic density scenario where the spectrum is shared by fewer vehicles and the inter-vehicle distance is larger. Meanwhile, BBCR showed the largest variance among all approaches as it tends to use the most spectrum resources for data transmission resulting in high co-channel interference to clusters sharing the same sub-channels. As for the two baseline approaches, Spatio also showed high data rate performance due to bandwidth over-provisioning as indicated by the light blue dash-dotted curves. The Uniform strategy, however, cannot satisfy the data rate requirement under all traffic densities because each cluster is only assigned a small portion of bandwidth given the limited spectrum resources. Note that, all strategies especially the comparison group showed periodic performance changes after each topology update.

5.6.3 Reliability performance analysis

Next, we analyze the reliability performance of all five approaches as shown in Fig. 5.6 and Table 5.3. QMAC guarantees the reliability requirement throughout the experiment under all traffic densities with the help of efficient error control and power selection. Over time, QMAC selects a suitable power level and error control scheme via the swap-regret minimization method. An action is selected with probability and the probability transition matrix is updated after each round. BBRC, Basic, and Spatio showed similar patterns in different scenarios, i.e., all of them showed periodic



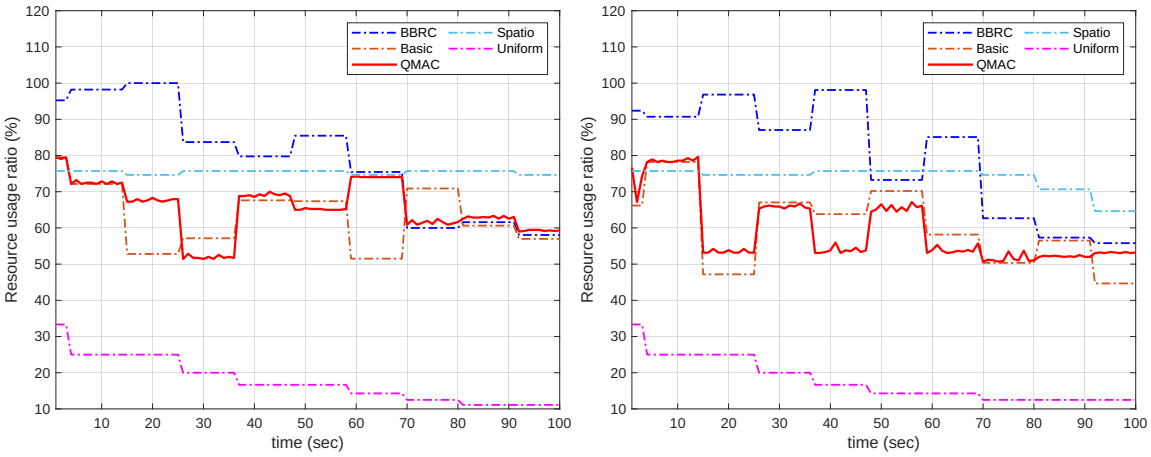
(a) High traffic density (b) Low traffic density

Figure 5.6: Reliability performance comparison under different traffic densities.

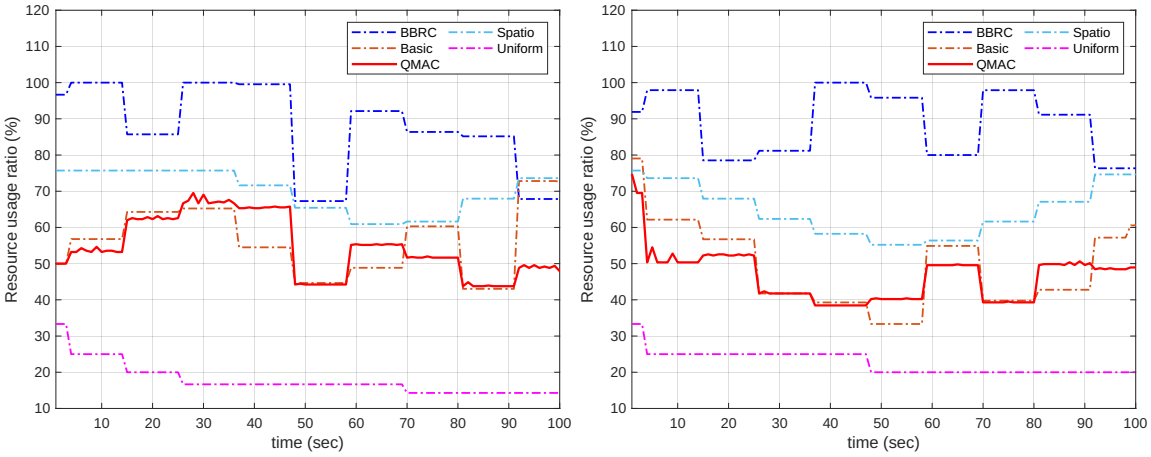
Table 5.3: Reliability Performance Comparison

Scheme	High	Medium	Low	Sparse
QMAC	**99.32%**	**99.41%**	**99.5%**	**99.51%**
Basic	97.91%	98.18%	98.43%	98.52%
BBRC	97.58%	97.83%	98.16%	98.5%
Spatio	97.87%	97.94%	97.82%	98.17%
Uniform	99.05%	99.08%	98.85%	99.03%

performance recovery and degradation after each topology update where the control decision is determined centrally. None of them guaranteed the reliability requirements although the performance is better under low traffic density due to less channel interference. The Uniform approach provided the second-best performance as each cluster was assigned non-overlapping sub-channels which contributes to higher SINR. Table 5.3 summarized the average reliability performance of all approaches throughout the experiments under four different traffic densities. Clearly, QMAC showed the best performance and guaranteed the 99% reliability requirement.



(a) High (b) Medium



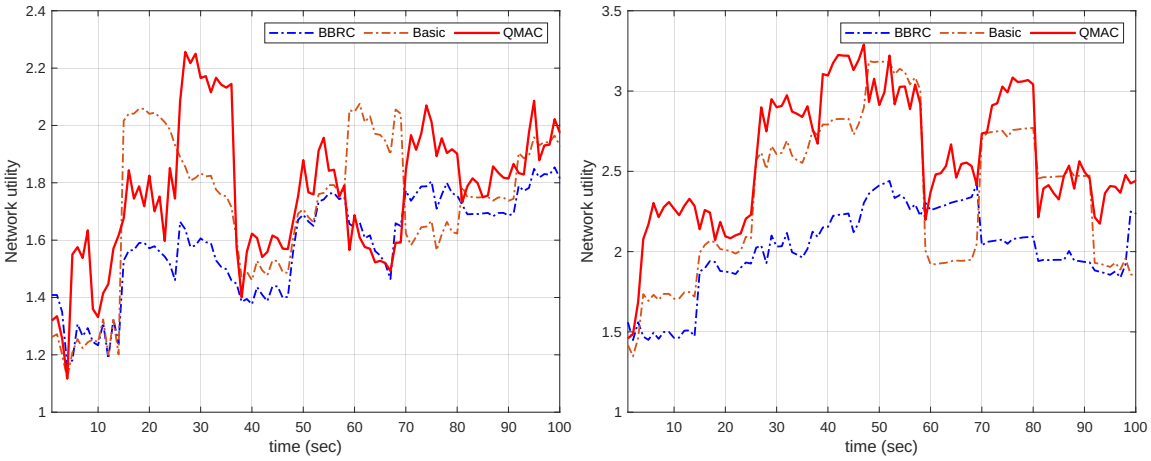
(c) Low (d) Sparse

Figure 5.7: Resource occupancy ratio comparison under different traffic densities.

5.6.4 Resource efficiency performance analysis

In this experiment, we analyze the resource usage efficiency of different approaches. We use the efficiency metric defined in (5.12) as a measurement of network utility and the spectrum occupancy ratio to characterize their performances as shown in Fig. 5.7 and Fig. 5.8, respectively.

We first analyze their occupancy ratio under different traffic densities. We use the equation $\frac{\text{number of used sub-channels}}{\text{total sub-channels}}$ to characterize the spectrum occupancy ratio. As shown in Fig. 5.7, the Uniform strategy gives the lowest occupancy ratio throughout



(a) High traffic density (b) Low traffic density

Figure 5.8: Network network utility performance comparison under different traffic densities.

the experiments as B_V is equally assigned to each cluster, i.e., the occupancy ratio remains constant at $1/M$. The value decreases over time as more clusters are formed. Similarly, the Spatio strategy also shows a more constant occupancy ratio as it assigns a fixed number of sub-channels to each cluster at each topology update. Spatio shows the second-highest occupancy ratio because the number of sub-channels assigned is calculated using the minimum applicable SINR value. BBRC shows the highest occupancy ratio due to its greedy bandwidth assignment strategy. On the other hand, QMAC and Basic present a lower occupancy ratio, up to 50% in high traffic density scenario and 60% in low traffic density scenario, compared to the BBRC and Spatio. This is because both QMAC and Basic continue to reduce the number of sub-channels after the first-round spectrum allocation. QMAC shows a higher occupancy ratio with some variations because, with the distributed control mechanism, each relay vehicle re-evaluates the current control strategy via channel sensing and selects or removes sub-channels to maintain the QoS requirements.

Furthermore, we compare the network utility gain achieved by the proposed solution as shown in Fig. 5.8. QMAC and Basic outperform BBRC under different traffic density scenarios as they satisfy the data rate requirement with fewer spectrum re-

sources. When the traffic density is high, although QMAC achieves a better network utility performance in general, Basic performs better under some circumstances. This is because QMAC used more sub-channels to achieve better SINR to maintain the reliability performance which is also reflected in other simulation results. For example, during the $15-25$ and $60-70$ time period, QMAC and Basic achieved similar data rate performance (QMAC performed slightly better) as shown in Fig. 5.5(a) but QMAC used more sub-channels as shown in Fig. 5.7. A similar pattern can be observed in lower traffic density scenarios as shown in Fig. 5.8(b).

5.7 Conclusion

In this chapter, we investigate the resource allocation problem in the advanced V2X application scenario. We aim to tradeoff between stringent QoS requirements and limited spectrum resources while addressing the varying channel condition issue caused by network mobility. In addition, we argue that in the advanced V2X scenario, a finer granularity usage of network resources is required. Thus, we propose to maximize resource efficiency by providing just enough resources to the network that can support the required QoS. To this end, we formulate a resource efficiency optimization problem and propose an efficient hybrid approach QMAC combining centralized and distributed control decision-making mechanisms.

Extensive simulations under different traffic densities are conducted to evaluate the proposed solution. The results show that QMAC achieves promising performances and outperforms the existing solution.

Chapter 6

Summary

In this thesis, we address the challenges of supporting stringent QoS requirements and improving network utility. We first proposed a novel protocol architecture SET which enables flexible protocol assembling via control function decomposition. A new concept called protocol control agent (PCA) is introduced in SET to enable in-network intelligence and enhanced adaptability. PCA leverages cross-layer network information and collaborations to support QoS requirements and improve resource efficiency. In addition, we considered the freshness of network information for designing new protocols based on SET architecture. Using the SET architecture as our design principle, we developed a range of scheduling and routing solutions and adopted a step-by-step strategy to verify their performances from a less constrained, wired network setting to a complicated and dynamic vehicular network.

In Chapter 2, we introduced a novel solution called delay-guaranteed scheduling and routing protocol (DSRP) for wired data networks. DSRP utilizes a distributed, delay-guaranteed, and congestion-aware network architecture. It employs cross-layer cooperation between link layer queue management and network layer routing to ensure packet-level end-to-end delay. Additionally, we propose a scheduling and routing algorithm based on renewal optimization, leading to significant improvements in goodput and network utility compared to existing approaches. In future works, it is worth investigating how to leverage more control functions such as congestion control and

active queue management to further improve network efficiency.

In Chapter 3, we introduced a delay-aware selective scheduling policy for downlink resource scheduling. This policy aims to guarantee the delay requirement of time-critical packets while optimizing network utility. We formulate a multi-objective optimization problem and propose a deep reinforcement learning framework to solve it. The proposed DRL framework shows faster convergence and significant reductions in average delay and delay outage drop rate. However, some issues in practice, e.g., wireless link dynamics, were not considered in the current work. In the future, more effort will be put into incorporating environmental dynamics to improve the scalability of our model.

In Chapter 4, we present the QoS-guaranteed Clustering and Routing Protocol (QCRP) in the vehicular network setting. QCRP utilizes global and local control agents to provide guaranteed QoS for V2X applications while adapting to network dynamics. Simulation experiments demonstrate the importance of both global and local control for maintaining end-to-end QoS performance over time. In future works, we can further explore how QCRP can be extended to multi-source scenarios and enable multiple applications with various QoS requirements to be served at the same time.

In Chapter 5, we focus on the resource allocation problem in vehicular networks following the same scenario explained in Chapter 4. We proposed a QoS-guaranteed Medium Access Control (QMAC) protocol which combines centralized spectrum allocation and distributed power and error control to optimize resource efficiency while guaranteeing QoS requirements. Extensive simulation experiments validate the effectiveness of QMAC in achieving high resource efficiency under different traffic densities. However, there are some other issues worth further research. For example, how QMAC can be improved by leveraging the vacancy spectrum resources. In addition, the service migration between different cells while not hurting the QoS performance is also worth exploring.

Overall, in this thesis, we introduced a new perspective of supporting stringent

QoS requirements and demonstrated the effectiveness of our solution in various network settings. Our work opens up new possibilities for research extensions and applications.

Appendix

6.1 Appendix for Chapter 2

Proof Suppose that for a given optimal policy π^*, there exists at least one queue Q_n that does not satisfy mean rate stability [97]. Thus we have

$$\lim_{t\to\infty} \frac{\mathbb{E}_{\pi^*}[Q_n(t)]}{t} \neq 0. \tag{6.1}$$

Then there exists a finite value υ such that $\mathbb{E}_{\pi^*}[Q_n(t)] \geq \delta t$ for $t \geq \upsilon$. Since $Q_n(0) = 0$, we can get the following inequality for $T > \upsilon$.

$$\begin{aligned} \sum_{t=0}^{T-1} \mathbb{E}_{\pi^*}[Q_n^2(t+1) - Q_n^2(t)] &= \mathbb{E}_{\pi^*}[Q_n^2(T-1)] \\ &\geq (\mathbb{E}_{\pi^*}[Q_n^2(T-1)]) \geq \delta^2(T-1)^2. \end{aligned} \tag{6.2}$$

Thus we have

$$\begin{aligned} &\lim_{t\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{u\in U} \mathbb{E}_{\pi}[-Q_u(t+1) - \nu_1(Q_u^2(t+1) - Q_u^2(t))] \\ &\leq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}_{\pi}[-Q_n(t+1) - \nu_1(Q_n^2(t+1) - Q_n^2(t))] \\ &\leq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} [-\delta t - \nu_1 \delta^2 (T-1)^2]. \end{aligned}$$

It is clear that the right-hand side of the above inequality tends to negative infinity, which contradicts the assumption in (6.1). Hence, all queues scheduled by an optimal policy of problem P1 satisfy the mean rate stability, which proves the first point in Proposition 1.

Let $Z_u(t) = \min[S_u(t)\hat{\eta}_{u,j} - \sum_{i\neq j, i\in Q_u(t)} D_{u,i}(t), Q_u(t)]$ for $u \in \mathcal{U}$. By (2.1),

$$Q_u(t+1) = Q_u(t) - Z_u(t) + A_u(t). \tag{6.3}$$

According to [99], when all queues scheduled by an optimal policy of problem P1 satisfy the mean rate stability, we have

$$\begin{aligned} &\lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\pi^*}\Big[\sum_{u\in U}[Q_u^2(t+1) - Q_u^2(t)]\Big] \\ &= \lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\pi^*}\sum_{u\in U}[(A_u(t) - Z_u(t)) \\ &\qquad\qquad \times (A_u(t) - Z_u(t) - 2Q_u(t))] = 0. \end{aligned} \tag{6.4}$$

Since $Q_u(t), u \in \mathcal{U}$ does not diverge, we have

$$\begin{aligned} &\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{u\in U}\mathbb{E}_{\pi}[-Q_u(t+1) - \nu_1(Q_u^2(t+1) - Q_u^2(t))] \\ &= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{u\in U}\mathbb{E}_{\pi}[-Q_u(t+1)]. \end{aligned}$$

Therefore, an optimal policy of problem P2 can minimize the average queue length while maximizing the average output gain, which proves the second point in Proposition 1.

6.2 Appendix for Chapter 3

Proof Suppose that for a given optimal policy π^*, there exists at least one queue Q_n that does not satisfy mean rate stability [95]. Thus we have

$$\lim_{t\to\infty}\frac{\mathbb{E}_{\pi^*}[Q_n(t)]}{t}\neq 0. \tag{6.5}$$

Then there exists a finite value υ such that $\mathbb{E}_{\pi^*}[Q_n(t)]\geq \delta t$ for $t\geq\upsilon$. Since $Q_n(0)=0$, we can get the following inequality for $T>\upsilon$.

$$\begin{aligned}\sum_{t=0}^{T-1}\mathbb{E}_{\pi^*}[Q_n^2(t+1)-Q_n^2(t)] &= \mathbb{E}_{\pi^*}[Q_n^2(T-1)]\\ &\geq (\mathbb{E}_{\pi^*}[Q_n^2(T-1)])\geq \delta^2(T-1)^2.\end{aligned} \tag{6.6}$$

Thus we have

$$\begin{aligned}&\lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{u\in U}\mathbb{E}_{\pi}[-Q_u(t+1)-\nu_1(Q_u^2(t+1)-Q_u^2(t))]\\ &\leq \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\pi}[-Q_n(t+1)-\nu_1(Q_n^2(t+1)-Q_n^2(t))]\\ &\leq \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}[-\delta t-\nu_1\delta^2(T-1)^2].\end{aligned}$$

It is clear that the right-hand side of the above inequality tends to negative infinity, which contradicts the assumption in (6.5). Hence, all queues scheduled by an optimal policy of problem P1 satisfy the mean rate stability, which proves the first point in Proposition 1.

Let $Z_u(t)=\min[S_u(t)\hat{\eta}_{u,j}-\sum_{i\neq j,i\in Q_u(t)}D_{u,i}(t),Q_u(t)]$ for $u\in\mathcal{U}$. By (3.7),

$$Q_u(t+1)=Q_u(t)-Z_u(t)+A_u(t). \tag{6.7}$$

According to [99], when all queues scheduled by an optimal policy of problem P1

satisfy the mean rate stability, we have

$$\begin{aligned}
&\lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\pi^*}\Big[\sum_{u\in U}[Q_u^2(t+1)-Q_u^2(t)]\Big] \\
&=\lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{\pi^*}\sum_{u\in U}[(A_u(t)-Z_u(t)) \\
&\qquad\qquad\times(A_u(t)-Z_u(t)-2Q_u(t))]=0.
\end{aligned} \tag{6.8}$$

Since $Q_u(t), u\in\mathcal{U}$ does not diverge, we have

$$\begin{aligned}
&\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{u\in U}\mathbb{E}_{\pi}[-Q_u(t+1)-\nu_1(Q_u^2(t+1)-Q_u^2(t))] \\
&=\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{u\in U}\mathbb{E}_{\pi}[-Q_u(t+1)].
\end{aligned}$$

Therefore, an optimal policy of problem P2 can minimize the average queue length while maximizing the average output gain, which proves the second point in Proposition 1.

6.3 Appendix for Chapter 4

In a BPSK system, denote E_b as the signal energy associated with each bit, and N_0 as the noise power level per hertz. Let $\gamma_b = E_b/N_0$, then the BER is given by

$$p_b = Q(\sqrt{2\gamma_b}), \tag{6.9}$$

where the Q function is defined as:

$$Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{x^2}{2}}dx. \tag{6.10}$$

Since E_b/N_0 is a normalized form of SNR, it can be represented by

$$\frac{E_b}{N_0} = \text{SNR} \cdot \frac{B}{f_b}, \tag{6.11}$$

where B is the channel bandwidth and f_b is the channel data rate. The channel model in (4.2) can be adopted to estimate SNR

$$\text{SNR} = \frac{P_t}{N_0} \delta d^{-\alpha} |h|^2. \tag{6.12}$$

Bringing (6.12) and (6.11) into (6.9), we can obtain the mapping between BER and distance

$$BER = f(d) = Q\left(\sqrt{2\frac{B}{f_b} \cdot \frac{P_t}{N_0} \delta d^{-\alpha} |h|^2}\right). \tag{6.13}$$

Bibliography

[1] 3GPP 5G NR (3GPP TS 38, Release 15).

[2] The Next Hyper-Connected Experience for All.

[3] Adaptive real-time routing in polynomial time. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 287–298. IEEE, 2019.

[4] Vamsi Addanki and Luigi Iannone. Moving a step forward in the quest for deterministic networks (detnet). In *2020 IFIP Networking Conference (Networking)*, pages 458–466. IEEE, 2020.

[5] Kunal Agrawal, Sanjoy Baruah, Zhishan Guo, Jing Li, and Sudharsan Vaidhun. Hard-real-time routing in probabilistic graphs to minimize expected delay. In *2020 IEEE Real-Time Systems Symposium (RTSS)*, pages 63–75. IEEE, 2020.

[6] Ahmed Al-Habob, Hina Tabassum, and Omer Waqar. Dynamic unicast-multicast scheduling for age-optimal information dissemination in vehicular networks. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1218–1223. IEEE, 2022.

[7] NGMN Alliance. 5g white paper. *Next generation mobile networks, white paper*, 1(2015), 2015.

[8] Eitan Altman. *Constrained Markov Decision Processes*. Kluwer Academic Publishers, USA, 1999.

[9] Matthew Andrews, Krishnan Kumaran, Kavita Ramanan, Alexander Stolyar, and Rajiv Vijayakumar. Scheduling in a queuing system with asynchronously varying service rates. *Probab. Eng. Inf. Sci.*, 18(2):191–217, Apr. 2004.

[10] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.*, 34(6):26–38, Nov. 2017.

[11] Shehzad Ali Ashraf, Ricardo Blasco, Hieu Do, Gabor Fodor, Congchi Zhang, and Wanlu Sun. Supporting vehicle-to-everything services by 5g new radio release-16 systems. *IEEE Communications Standards Magazine*, 4(1):26–32, 2020.

[12] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[13] Muddasar Ayyub, Alma Oracevic, Rasheed Hussain, Ammara Anjum Khan, and Zhongshan Zhang. A comprehensive survey on clustering in vehicular networks: Current solutions and future challenges. *Ad Hoc Networks*, 124:102729, 2022.

[14] Jeongmin Bae, Joohyun Lee, and Song Chong. Learning to schedule network resources throughput and delay optimally using Q+-learning. *IEEE/ACM Trans. Netw.*, 29(2):750–763, Apr. 2021.

[15] Luca Barbieri, Stefano Savazzi, Mattia Brambilla, and Monica Nicoli. Decentralized federated learning for extended sensing in 6g connected vehicles. *Veh. Commun.*, 33:100396, 2022.

[16] Sanjoy Baruah. Rapid routing with guaranteed delay bounds. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 13–22. IEEE, 2018.

[17] Amina BENGAG, Asmae Bengag, and Mohamed EL Boukhari. Enhancing gpsr routing protocol based on velocity and density for real-time urban scenario. In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–5. IEEE, 2020.

[18] Jitendra Bhatia, Ridham Dave, Heta Bhayani, Sudeep Tanwar, and Anand Nayyar. Sdn-based real-time urban traffic analysis inVANET environment. *Comput. Commun.*, 149:162–175, 2020.

[19] Massieh Kordi Boroujeny, Brian L Mark, and Yariv Ephraim. Stochastic traffic regulator for end-to-end network delay guarantees. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[20] Andres Burbano-Abril, Brian McCarthy, Miguel Lopez-Guerrero, Victor Rangel, and Aisling O'Driscoll. Mcs adaptation within the cellular V2X sidelink. In *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 111–117. IEEE, 2021.

[21] Lin Cai, Jianping Pan, Wenjun Yang, Xiangyu Ren, and Xuemin Shen. Self-evolving and transformative (SET) protocol architecture for 6G. *IEEE Wireless Commun.*, pages 1–12, early access, Aug. 22, 2022. doi: 10.1109/MWC.003.2200022.

[22] Liu Cao, Sumit Roy, and Hao Yin. Resource allocation in 5G platoon communication: Modeling, analysis and optimization. *IEEE Trans. Veh. Commun.*, 72(4):5035–5048, 2022.

[23] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53, 2016.

[24] Guner D. Celik, Long B. Le, and Eytan Modiano. Dynamic server allocation over time-varying channels with switchover delay. *IEEE Trans. Inf. Theory*, 58(9):5856–5877, Sep. 2012.

[25] Guanhua Chai, Weihua Wu, Qinghai Yang, Meng Qin, Yan Wu, and F Richard Yu. Platoon partition and resource allocation for ultra-reliable V2X networks. *IEEE Trans. Veh. Commun.*, 2023.

[26] Shih-Hung Chang, Huan Chen, and Bo-Chao Cheng. Time-predictable routing algorithm for time-sensitive networking: Schedulable guarantee of time-triggered streams. *Computer Communications*, 172:183–195, 2021.

[27] Shih-Hung Chang, Huan Chen, and Bo-Chao Cheng. Time-predictable routing algorithm for time-sensitive networking: Schedulable guarantee of time-triggered streams. *Computer Communications*, 172:183–195, 2021.

[28] Twinkle Chatterjee, Raja Karmakar, Georges Kaddoum, Samiran Chattopadhyay, and Sandip Chakraborty. A survey of VANET/V2X routing from the perspective of non-learning-and learning-based approaches. *IEEE Access*, 10:23022–23050, 2022.

[29] Chen Chen, Lei Liu, Tie Qiu, Dapeng Oliver Wu, and Zhiyuan Ren. Delay-aware grid-based geographic routing in urbanVANETs: A backbone approach. *IEEE/ACM Trans. Netw.*, 27(6):2324–2337, 2019.

[30] Jiayin Chen, Peng Yang, Qiang Ye, and Xu Li. Learning-based proactive resource allocation for delay-sensitive packet transmission. *IEEE Trans. Cogn. Commun. Netw.*, 7(2):675–688, Jun. 2021.

[31] Yi Chen, Xuan Wang, and Lin Cai. On achieving fair and throughput-optimal scheduling for TCP flows in wireless networks. *IEEE Transactions on wireless communications*, 15(12):7996–8008, 2016.

[32] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). Technical report, 2003.

[33] Alexander Clemm and Toerless Eckert. High-precision latency forwarding over packet-programmable networks. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

[34] Ioan. Sorin Comsa, Sijing Zhang, Mehmet Emin Aydin, Pierre Kuonen, Yao Lu, Ramona Trestian, and Gheorghita Ghinea. Towards 5G: A reinforcement learning-based scheduling solution for data traffic management. *IEEE Trans. Netw. Serv. Manag.*, 15(4):1661–1675, Dec. 2018.

[35] Carlo Curino, Djellel E Difallah, Chris Douglas, Subru Krishnan, Raghu Ramakrishnan, and Sriram Rao. Reservation-based scheduling: If you're late don't blame us! In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14, 2014.

[36] Frank E Curtis, Michael J O'Neill, and Daniel P Robinson. Worst-case complexity of an sqp method for nonlinear equality constrained stochastic optimization. *Mathematical Programming*, pages 1–53, 2023.

[37] Mohammed Elaryh Makki Dafalla, Rania A Mokhtar, Rashid A Saeed, Hesham Alhumyani, S Abdel-Khalek, and Mashael Khayyat. An optimized link state routing protocol for real-time application over vehicular ad-hoc network. *Alexandria Engineering Journal*, 61(6):4541–4556, 2022.

[38] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.*, 18(4):577–601, Aug. 2014.

[39] Nicolò Decarli, Anna Guerra, Caterina Giovannetti, Francesco Guidi, and Barbara M Masini. V2x sidelink localization of connected automated vehicles. *IEEE J. Sel. Areas Commun.*, 2023.

[40] Atilla Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE Trans. Netw.*, 15(6):1333–1344, Dec. 2007.

[41] Andres Ferragut and Fernando Paganini. Network resource allocation for users with multiple connections: Fairness and stability. *IEEE/ACM Trans. Netw.*, 22(2):349–362, Apr. 2014.

[42] Swati Gangopadhyay and Vinod Kumar Jain. A position-based modified olsr routing protocol for flying ad hoc networks. *IEEE Trans. Veh. Technol.*, 2023.

[43] Mario H Castañeda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan Şahin, and Apostolos Kousaridas. A tutorial on 5GNRV2X communications. *IEEE Commun. Surv. Tutorials*, 23(3):1972–2026, 2021.

[44] Michele Garetto and Don Towsley. Modeling, simulation and measurements of queuing delay under long-tail Internet traffic. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):47–57, 2003.

[45] Leonidas Georgiadis, Michael J. Neely, and Leandros Tassiulas. Resource allocation and cross-layer control in wireless networks. *Found. Trends Netw.*, 1(1):1–149, Oct. 2006.

[46] Hongyu Gong, Luoyi Fu, Xinzhe Fu, Lutian Zhao, Kainan Wang, and Xinbing Wang. Distributed multicast tree construction in wireless sensor networks. *IEEE Transactions on Information Theory*, 63(1):280–296, 2016.

[47] Chi Guo, Cong Wang, Lin Cui, Qiuzhan Zhou, and Juan Li. Radio resource management for c-v2x: From a hybrid centralized-distributed scheme to a distributed scheme. *IEEE J. Sel. Areas Commun.*, 41(4):1023–1034, 2023.

[48] Chongtao Guo, Le Liang, and Geoffrey Ye Li. Resource allocation for veh. commun. with low latency and high reliability. *IEEE Transactions on Wireless Communications*, 18(8):3887–3902, 2019.

[49] Marina Gutiérrez, Astrit Ademaj, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. Self-configuration of ieee 802.1 tsn networks. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[50] Marina Gutiérrez, Astrit Ademaj, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. Self-configuration of IEEE 802.1 TSN networks. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[51] Nikola Gvozdiev, Stefano Vissicchio, Brad Karp, and Mark Handley. On low-latency-capable topologies, and their impact on the design of intra-domain routing. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 88–102, 2018.

[52] Long Hai, Qinghua Gao, Jie Wang, He Zhuang, and Ping Wang. Delay-optimal back-pressure routing algorithm for multihop wireless networks. *IEEE Transactions on Vehicular Technology*, 67(3):2617–2630, 2017.

[53] Long Hai, Qinghua Gao, Jie Wang, He Zhuang, and Ping Wang. Delay-optimal back-pressure routing algorithm for multihop wireless networks. *IEEE Trans. Veh. Technol.*, 67(3):2617–2630, Nov. 2017.

[54] Saqib Hakak, Thippa Reddy Gadekallu, Praveen Kumar Reddy Maddikunta, Swarna Priya Ramu, M Parimala, Chamitha De Alwis, and Madhusanka Liyan-

age. Autonomous vehicles in 5G and beyond: A survey. *Veh. Commun.*, 39:100551, 2023.

[55] Mehdi Harounabadi, Dariush Mohammad Soleymani, Shubhangi Bhadauria, Martin Leyh, and Elke Roth-Mandutz. V2x in 3gpp standardization: Nr sidelink in release-16 and beyond. *IEEE Communications Standards Magazine*, 5(1):12–21, 2021.

[56] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. *Comput. ence*, 2016.

[57] Jianhua He, Kun Yang, and Hsiao-Hwa Chen. 6g cellular networks and connected autonomous vehicles. *IEEE Network*, 35(4):255–261, 2020.

[58] Jingfei Huang, Yang Yang, and Gang He. Deep reinforcement learning-based dynamic spectrum access for D2D communication underlay cellular networks. *IEEE Commun. Lett.*, 25(8):2614–2618, Aug. 2021.

[59] Tongyi Huang, Wu Yang, Jun Wu, Jin Ma, Xiaofei Zhang, and Daoyin Zhang. A survey on green 6G network: Architecture and technologies. *IEEE access*, 7:175758–175768, 2019.

[60] Zhiming Huang, Kaiyang Liu, and Jianping Pan. End-to-end congestion control as learning for unknown games with bandit feedback. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 327–338. IEEE, 2023.

[61] Rasheed Hussain and Sherali Zeadally. Autonomous cars: research results, issues, and future challenges. *IEEE Commun. Surv. Tutor*, 21(2):1275–1313, 2018.

[62] Md Mahmudul Islam, Muhammad Toaha Raza Khan, Malik Muhammad Saad, and Dongkyun Kim. Software-defined vehicular network (sdvn): A survey on architecture and routing. *Journal of Systems Architecture*, 114:101961, 2021.

[63] Byeong-Hoon Jang, Sunghwa Son, and Kyung-Joon Park. Deadline-aware routing with probabilistic delay guarantee in cyber-physical systems. In *2017 International Conference on Information Networking (ICOIN)*, pages 51–53. IEEE, 2017.

[64] Wei Jiang, Bin Han, Mohammad Asif Habibi, and Hans Dieter Schotten. The road towards 6G: A comprehensive survey. *IEEE Open Journal of the Communications Society*, 2:334–366, 2021.

[65] Zhenzhen Jiao, Cheng Li, and Hussein T Mouftah. Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey. *IEEE Wireless Commun.*, 23(1):102–110, Mar. 2016.

[66] Zhenzhen Jiao, Baoxian Zhang, Cheng Li, and Hussein T Mouftah. Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey. *IEEE Wireless Communications*, 23(1):102–110, 2016.

[67] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.

[68] Ajmal Khan, Afsah Abid Siddiqui, Farman Ullah, Muhammad Bilal, Md Jalil Piran, and Houbing Song. Vp-cast: Velocity and position-based broadcast suppression forVANETs. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[69] Taehyoung Kim, Younsun Kim, Minchae Jung, and Hyukmin Son. Intelligent partial sensing based autonomous resource allocation forNRv2x. *IEEE Internet Things J.*, 2023.

[70] Xiaolong Lan, Yi Chen, and Lin Cai. Throughput-optimal H-QMW scheduling for hybrid wireless networks with persistent and dynamic flows. *IEEE Transactions on Wireless Communications*, 19(2):1182–1195, 2019.

[71] Long Bao Le, Eytan Modiano, and Ness B Shroff. Optimal control of wireless networks with finite buffers. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[72] J.W. Lee, R.R. Mazumdar, and N.B. Shroff. Opportunistic power scheduling for dynamic multi-server wireless systems. *IEEE Trans. Wireless Commun.*, 5(6):1506–1515, Jun. 2006.

[73] E. Leonardi, M. Mellia, and F. Neri. Bounds on average delays and queue size averages and variances in input-queued cell-based switches. In *Proc IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pages 1095–1103, Aug. 2001.

[74] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*, 2018.

[75] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. DetNet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*, 2018.

[76] Zhuhui Li, Liang Zhao, Geyong Min, Ahmed Y Al-Dubai, Ammar Hawbani, Albert Y Zomaya, and Chunbo Luo. Reliable and scalable routing under hybrid sdvn architecture: A graph learning based method. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[77] Le Liang, Hao Ye, and Geoffrey Ye Li. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE J. Sel. Areas Commun.*, 37(10):2282–2292, 2019.

[78] Wenjie Liu, Haixia Zhang, Hui Ding, and Dongfeng Yuan. Delay and energy minimization for adaptive video streaming: A joint edge caching, computing and power allocation approach. *IEEE Transactions on Vehicular Technology*, 71(9):9602–9612, 2022.

[79] M. Marsan, E. Leonardi, and F. Neri. On the stability of local scheduling policies in networks of packet switches with input queues. *IEEE J. Sel. Areas Commun.*, 21(4):642–655, May. 2003.

[80] MATLAB. *MATLAB Optimization Toolbox (2024a)*. The MathWorks Inc., Natick, Massachusetts, 2024.

[81] M.W. McConley, B.D. Appleby, M.A. Dahleh, and E. Feron. A computationally efficient Lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees. *IEEE Trans. Autom. Control*, 45(1):33–49, Jan. 2000.

[82] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving %100 throughput in an input-queued switch. *IEEE Trans. Commun.*, 47(8):1260–1267, Aug. 1999.

[83] Deep Medhi and Karthik Ramasamy. Chapter 19 Circuit-Switching: Hierarchical and Dynamic Call Routing. In Deep Medhi and Karthik Ramasamy, editors, *Network Routing (Second Edition)*, The Morgan Kaufmann Series in Networking, pages 646–672. Morgan Kaufmann, Boston, second edition edition, 2018.

[84] Xiaoyu Mo, Zhiyu Huang, Yang Xing, and Chen Lv. Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. *IEEE Trans. Intell. Transp. Syst*, 23(7):9554–9567, 2022.

[85] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali. Routing without routes: The backpressure collection protocol. In *Proc Inf. Proc. Sensor Netw. (IPSN)*, pages 279–290, Apr. 2010.

[86] Hamed Mosavat-Jahromi, Yue Li, Lin Cai, and Lei Lu. Nc–mac: a distributed mac protocol for reliable beacon broadcasting in v2x. *IEEE Trans. Veh. Commun.*, 70(6):6044–6057, 2021.

[87] Ankur Nahar and Debasis Das. Metalearn: Optimizing routing heuristics with a hybrid meta-learning approach in vehicular ad-hoc networks. *Ad Hoc Networks*, 138:102996, 2023.

[88] T. Nakamura. 5G evolution and 6G. In *2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–1, 2020.

[89] Zhaojun Nan, Yunjian Jia, Zhi Ren, Zhengchuan Chen, and Liang Liang. Delay-aware content delivery with deep reinforcement learning in internet of vehicles. *IEEE Trans. Intell. Transp. Syst.*, 23(7):8918–8929, 2021.

[90] Ahmed Nasrallah, Akhilesh S Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research. *IEEE Communications Surveys & Tutorials*, 21(1):88–145, 2018.

[91] Ahmed Nasrallah, Akhilesh S Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research. *IEEE Communications Surveys & Tutorials*, 21(1):88–145, 2018.

[92] Valery Naumov and Thomas R Gross. Connectivity-aware routing (car) in vehicular ad-hoc networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Comput. Commun.*, pages 1919–1927. IEEE, 2007.

[93] Michael J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *IEEE J. Sel. Areas Commun.*, 24(8):1489–1501, Aug. 2006.

[94] Michael J. Neely. Stochastic network optimization with application to communication and queueing systems. *Synth. Lectures Commun. Netw.*, 3(1):1–211, Jan. 2010.

[95] Michael J Neely. Delay-based network utility maximization. *IEEE/ACM Trans. Netw.*, 21(1):41–54, Apr. 2012.

[96] Michael J Neely. Dynamic optimization and learning for renewal systems. *IEEE Transactions on Automatic Control*, 58(1):32–46, 2012.

[97] Michael J Neely. Fast learning for renewal optimization in online task scheduling. *arXiv preprint arXiv:2007.09532*, 2020.

[98] Michael J Neely. Fast learning for renewal optimization in online task scheduling. *J. Mach. Learn. Res.*, 22:279–1, 2021.

[99] Michael J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE J. Sel. Areas Commun.*, 23(1):89–103, Jan. 2005.

[100] Michael J Neely, Eytan Modiano, and Chih-Ping Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions On Networking*, 16(2):396–409, 2008.

[101] Michael J. Neely, Eytan Modiano, and Chih-Ping Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Trans. Netw.*, 16(2):396–409, Apr. 2008.

[102] Michael J Neely, Eytan Modiano, and Charles E Rohrs. Dynamic power allocation and routing for time varying wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 1, pages 745–755. IEEE, 2003.

[103] Yu E Nesterov and Michael J Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations research*, 22(1):1–42, 1997.

[104] Bach Long Nguyen, Duy Trong Ngo, Nguyen H Tran, Minh N Dao, and Hai L Vu. Dynamic v2i/v2v cooperative scheme for connectivity and through-

put enhancement. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1236–1246, 2020.

[105] Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Mohammad Omar Khyam, Jianhua He, Dirk Pesch, Klaus Moessner, Walid Saad, and H Vincent Poor. 6G for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities. *Proceedings of the IEEE*, 110(6):712–734, 2022.

[106] SungHeun Oh, Ara Khil, and SeungMin Yang. A modified least-laxity first scheduling algorithm for reducing context switches on multiprocessor systems. *J. Comput. Sys. Theory*, 30(12):68–77, Feb. 2003.

[107] Mohammad Parvini, Mohammad Reza Javan, Nader Mokari, Bijan Abbasi, and Eduard A Jorswieck. Aoi-aware resource allocation for platoon-based c-v2x networks via multi-agent multi-task reinforcement learning. *IEEE Trans. Veh. Commun.*, 2023.

[108] Adam Paszke, Sam Gross, Francisco Massa, and Adam Lerer. Pytorch: An imperative style, high-performance deep learning library. In *Proc Neural Inf. Proc. Sys. (NIPS)*, pages 8026–8037, 2019.

[109] Weijing Qi, Björn Landfeldt, Qingyang Song, Lei Guo, and Abbas Jamalipour. Traffic differentiated clustering routing in DSRC and C-V2X hybrid vehicular networks. *IEEE Trans. Vehicular Tech.*, 69(7):7723–7734, 2020.

[110] Nitin Singh Rajput, Upendra Singh, Amit Dua, Neeraj Kumar, Joel JPC Rodrigues, Sheetal Sisodia, Mohammed Elhoseny, and Yahya Lakys. Amalgamating vehicular networks with vehicular clouds, ai, and big data for next-generation its services. *IEEE Trans. Intell. Transp. Syst.*, 2023.

[111] Xiangyu Ren, Lin Cai, and Pooria Seyed Eftetahi. Qos-guaranteed clustering and routing protocol for extended sensor sharing in vehicular networks.

In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pages 2991–2996. IEEE, 2023.

[112] Xiangyu Ren, Jiequ Ji, and Lin Cai. Delay laxity-based scheduling with double-deep q-learning for time-critical applications. In *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2022.

[113] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[114] Malik Muhammad Saad, Muhammad Ashar Tariq, Junho Seo, and Dongkyun Kim. An overview of 3GPP release 17 & 18 advancements in the context of V2X technology. In *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 057–062. IEEE, 2023.

[115] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A vision of 6G wireless systems: Applications, trends, technologies, and open research problems. *IEEE network*, 34(3):134–142, 2019.

[116] Anirudha Sahoo and Shanti Chilukuri. DGRAM: a delay guaranteed routing and MAC protocol for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(10):1407–1423, 2010.

[117] Khabaz Sehla, Thi Mai Trang Nguyen, Guy Pujolle, and Pedro Braconnot Velloso. Resource allocation modes in c-v2x: from LTE-V2X to 5g-v2x. *IEEE Internet Things J.*, 9(11):8291–8314, 2022.

[118] Wen-Di Shen and Hung-Yu Wei. Distributed V2X sidelink communications with receiver grant mac design. *IEEE Trans. Veh. Commun.*, 71(5):5415–5429, 2022.

[119] Prashant Kumar Shrivastava and LK Vishwamitra. Comparative analysis of proactive and reactive routing protocols inVANET environment. *Measurement: Sensors*, 16:100051, 2021.

[120] Benjamin Sliwa, Robert Falkenberg, and Christian Wietfeld. Towards cooperative data rate prediction for future mobile and vehicular 6g networks. In *2020 2nd 6G Wireless Summit (6G SUMMIT)*, pages 1–5. IEEE, 2020.

[121] Jie Song, Yi Wu, Zhexin Xu, and Xiao Lin. Research on car-following model based on sumo. In *The 7th IEEE/International Conference on Advanced Infocomm Technology*, pages 47–55. IEEE, 2014.

[122] Ankita Srivastava, Arun Prakash, and Rajeev Tripathi. Location based routing protocols in vanet: Issues and existing solutions. *Vehicular Communications*, 23:100231, 2020.

[123] Ashutosh Srivastava, Soumyadeep Datta, Sanjay Goyal, Umer Salim, Wasif Jawad Hussain, Pei Liu, Shivendra Panwar, Ravikumar Pragada, and Pascal Adjakple. Enhanced distributed resource selection and power control for high frequencyNRV2X sidelink. *IEEE Access*, 2023.

[124] Yujie Tang, Nan Cheng, Wen Wu, Miao Wang, Yanpeng Dai, and Xuemin Shen. Delay-minimization routing for heterogeneousVANETs with machine learning based mobility prediction. *IEEE Trans. Vehicular Tech.*, 68(4):3967–3979, 2019.

[125] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Autom. Control*, 37(12):1936–1948, Dec. 1992.

[126] Leandros Tassiulas and Anthony Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39(2):466–478, 1993.

[127] Ioannis Tomkos, Dimitrios Klonidis, Evangelos Pikasis, and Sergios Theodoridis. Toward the 6G network era: Opportunities and challenges. *IT Professional*, 22(1):34–38, 2020.

[128] Ozan K Tonguz, Nawaporn Wisitpongphan, and Fan Bai. Dv-cast: A distributed vehicular broadcast protocol for vehicular ad hoc networks. *IEEE Wireless Commun.*, 17(2):47–57, 2010.

[129] Jamal Toutouh, José García-Nieto, and Enrique Alba. Intelligent olsr routing protocol optimization forVANETs. *IEEE Trans. Veh. Technol.*, 61(4):1884–1894, 2012.

[130] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[131] Pengfei Wang, Boya Di, Hongliang Zhang, Kaigui Bian, and Lingyang Song. Platoon cooperation in cellular V2X networks for 5G and beyond. *IEEE Transactions on Wireless Communications*, 18(8):3919–3932, 2019.

[132] Xinyu Wu, Dan Wu, and Eytan Modiano. Overload balancing in single-hop networks with bounded buffers. In *2022 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2022.

[133] Binghao Yan, Qinrang Liu, JianLiang Shen, Dong Liang, Bo Zhao, and Ling Ouyang. A survey of low-latency transmission strategies in software defined networking. *Computer Science Review*, 40:100386, 2021.

[134] Mao Yang, Yong Li, Depeng Jin, Lieguang Zeng, Xin Wu, and Athanasios V Vasilakos. Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, 20:4–18, 2015.

[135] Ibrar Yaqoob, Iftikhar Ahmad, Ejaz Ahmed, Abdullah Gani, Muhammad Imran, and Nadra Guizani. Overcoming the key challenges to establishing vehicular communication: Is SDN the answer? *IEEE Commun. Mag.*, 55(7):128–134, 2017.

[136] Sofiane Zaidi, Salim Bitam, and Abdelhamid Mellouk. Enhanced adaptive sub-packet forward error correction mechanism for video streaming in VANET. In *IEEE Glob.*, pages 1–6, Dec. 2016.

[137] Fenghui Zhang, Michael Mao Wang, Xuecai Bao, and Weirong Liu. Centralized resource allocation and distributed power control for noma-integratedNRv2x. *IEEE Internet Things J.*, 8(22):16522–16534, 2021.

[138] Hui Zhang, Xinming Zhang, and Dan Keun Sung. A fast, reliable, opportunistic broadcast scheme with mitigation of internal interference inVANETs. *IEEE Trans. Mob. Comput.*, 2021.

[139] Ticao Zhang, Shuyi Shen, Shiwen Mao, and Gee-Kung Chang. Delay-aware cellular traffic scheduling with deep reinforcement learning. In *Proc IEEE Global Commun. Conf. (GLOBECOM)*, pages 1–6, Jan. 2020.

[140] Xin Ming Zhang, Yue Zhang, Fan Yan, and Athanasios V Vasilakos. Interference-based topology control algorithm for delay-constrained mobile ad hoc networks. *IEEE Trans. Mob. Comput.*, 14(4):742–754, 2014.

[141] Liang Zhao, Zhenguo Bi, Ammar Hawbani, Keping Yu, Yan Zhang, and Mohsen Guizani. Elite: An intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks. *IEEE Transactions on Mobile Computing*, 2022.

[142] Liang Zhao, Guangjie Han, Zhuhui Li, and Lei Shu. Intelligent digital twin-based software-defined vehicular networks. *IEEE Network*, 34(5):178–184, 2020.

[143] Yi Zhong, Tony QS Quek, and Xiaohu Ge. Heterogeneous cellular networks with spatio-temporal traffic: Delay analysis and scheduling. *IEEE Journal on Selected Areas in Communications*, 35(6):1373–1386, 2017.